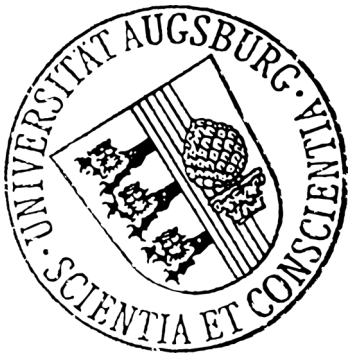
Dissertation

Automatic Object Annotations From Weakly Labeled Images

Christian X. Ries



Department of Computer Science

University of Augsburg

Advisor:	Prof. Dr. Rainer Lienhart
Reviewers:	Prof. Dr. Rainer Lienhart
	Prof. Dr. Bernhard Möller
Thesis Defense:	August 13, 2014

ries@informatik.uni-augsburg.de

Abstract

In the field of computer vision, algorithms for image classification, object detection, and image retrieval are among the principal research topics. Such algorithms usually benefit from object annotations in training images indicating the locations of desired objects.

In this thesis, an approach for automatically determining object annotations in the form of bounding boxes is presented. The goal of the approach is to devise bounding boxes only based on binary global image labels. That is, the only given information we want to exploit is whether or not a given training image shows the desired object. In other words, we are only given a (positive) set of images containing the object and a (negative) set of images not containing the object. Our task is then to deduce the locations of the wanted object within the positive images without further knowledge.

The approach presented in this thesis is a two-stage process. In the first stage, a statistical feature model is created which determines visual features which are likely to be indicative for the desired object. After determining such features in the form of pixel colors and gradient features, we deduce a set of positive pixels and ultimately one or multiple bounding boxes for each positive image. We experimentally show that these boxes often already exclude a considerable amount of background from positive images.

In the second stage, we further improve our bounding box estimations using a machine learning algorithm which is based on linear structural Support Vector Machines. The unknown actual object bounding boxes are modeled by latent variables which are re-estimated iteratively in our learning algorithm based on the Convex-Concave Procedure (CCCP). Thus, the final output of our algorithm are new estimations for the bounding box locations. We also provide an in-depth explanation for the latent structural SVM learning algorithm.

Quantitative evaluations of all components of our approach are performed on three publicly available datasets and the results are thoroughly discussed. In additional experiments we examine the usefulness of our automatically determined bounding boxes for image retrieval and classification.

Danksagung

Ich bedanke mich bei Prof. Dr. Rainer Lienhart, der mir diese Arbeit ermöglicht und sie gewissenhaft und kompetent betreut hat. Ebenso bedanke ich mich bei Prof. Dr. Bernhard Möller, der das Zweitgutachten übernommen hat. Außerdem danke ich Prof. Dr. Jörg Hähner, der sich als Prüfer zur Verfügung gestellt hat.

Meinen Kollegen und ehemaligen Kollegen Fabian Richter, Stefan Romberg, Gregor van den Boogaart, Nicolas Cebron, Christian Eggert, Thomas Greif, Eva Hörster, Christoph Lassner, Johannes Schels und Dan Zecha bin ich sehr dankbar für die angenehme und produktive Zusammenarbeit und viele aufschlussreiche Diskussionen.

Besonderer Dank gilt meiner Familie, insbesondere meinen Eltern Hannelore Ries und Dr. Franz Xaver Ries, die mir mein Studium ermöglicht und mich immer unterstützt haben. Für moralische Unterstützung danke ich Swetlana Martens.

CONTENTS

1. Introduction 11
1.1 Problem Definition 11
1.2 Assumptions 13
1.3 Overview 13

2. Related Work 16
2.1 Previous Work 16
2.2 Work on Related Problems 17

3. Datasets 21
3.1 Brand Logos 21
3.2 Flowers 22
3.3 3D Object Categories 24
3.4 Bikini 25
3.5 Negative Sets 26

4. Discriminative Feature Model 28
4.1 Statistical Model 30
4.2 Confidence Intervals 32
4.3 Global Threshold 35

5. Regions of Interest Based on Discriminative Model 38
5.1 Gaussian Bounding Boxes 38
5.1.1 Binary Maps 38
5.1.2 Bounding Boxes on Positive Pixels 39
5.2 Quality of Bounding Boxes 40

6. Color Model 43
6.1 Initial Model 43
6.2 Adding Spatially and Chromatically Related Colors 44

6.3 Evaluation of Color Model . . . 47
6.3.1 Parameter Evaluation on Brand Logos . . . 47
6.3.2 Evaluation on Flowers Dataset . . . 52
6.3.3 Application to Skin Detection . . . 56
6.4 Evaluation of Color Model for Automatic Object Annotation . 58
6.4.1 Brand Logos . . . 60
6.4.2 Flowers . . . 60
6.4.3 3D Object Categories . . . 62
6.4.4 Bikini . . . 64

7. HOG Model . . . 66
7.1 Multi-Scale HOG . . . 66
7.2 Feature Quantization . . . 69
7.3 Discriminative Model . . . 69
7.4 Evaluation . . . 71
7.4.1 Brand Logos . . . 71
7.4.2 Flowers . . . 74
7.4.3 3D Object Categories . . . 75

8. Combined Model . . . 79
8.1 Color-and-Hog Model . . . 79
8.2 Evaluation . . . 80
8.2.1 Brand Logos . . . 80
8.2.2 Flowers . . . 84
8.2.3 3D Object Categories . . . 86

9. Multiple Bounding Boxes . . . 88
9.1 Heuristic for Multiple Bounding Boxes . . . 88
9.2 Additional Filters and Merging . . . 91
9.3 Hough Voting . . . 91
9.4 Evaluation on Multiple Instances . . . 93
9.4.1 Brand Logos . . . 94
9.4.2 Flowers . . . 97
9.4.3 3D Object Categories . . . 97

10. Improving Initial Bounding Boxes by Latent Structural SVM . . . 100
10.1 Linear SVM . . . 102
10.1.1 Hard Margin SVM . . . 102

10.1.2 Soft Margin SVM . . . 105
10.2 Linear SVM for Structured Output . . . 106
10.2.1 Formulation of Structured Problem . . . 108
10.3 Adding latent variables . . . 111
10.3.1 Prediction With Latent Variables . . . 111
10.3.2 Regularized Risk for Structural SVM . . . 112
10.3.3 Optimization Problem for Latent Variables . . . 113
10.4 Training algorithm . . . 114
10.5 Implementation . . . 116
10.5.1 Example Representation and Latent Variables . . . 117
10.5.2 Loss Function and Model Vector Learning . . . 118
10.5.3 Efficient Implementation of Rectangle Search . . . 121
10.5.4 Search Space for Inference of Latent Variables . . . 122
10.5.5 Post-Processing . . . 125
10.6 Evaluation . . . 127
10.6.1 Brand Logos . . . 127
10.6.2 Flowers . . . 132
10.6.3 3D Object Categories . . . 133
10.7 Discussion . . . 137

11. Evaluation of Applications for Automatic Annotations . . . 140
11.1 Evaluation for Image Retrieval . . . 140
11.2 Evaluation for Classification . . . 142

12. Summary and Conclusion . . . 146
12.1 Summary . . . 146
12.2 Outlook . . . 147
12.3 Conclusion . . . 148

Appendix 150

A. SVM Optimization Problem . . . 151
A.1 Lagrange Multiplier Method . . . 152
A.1.1 Single Equality Constraint . . . 152
A.1.2 Multiple Equality Constraints . . . 153
A.1.3 Inequality Constraints . . . 155
A.2 Dual Problem Formulation . . . 156

LIST OF TERMS

Automatic annotation An object annotation which is derived only from global image labels without any further previous knowledge.

Bounding box A rectangle tightly enclosing an object instance.

Negative feature A visual feature which is not indicative for a desired object class, i.e. which is common in image background or negative images.

Negative pixel A pixel not belonging to an instance of the desired object class, i.e. a background pixel.

Negative image An image not contaning any object instance.

Object annotation A bounding box defining the location of an object instance within a positive image.

Object class A category of objects such as "car" or "DHL brand logo". The *desired* or *wanted* object class is the class for which we want to determine automatic annotations.

Object instance One instance of an object class occurring within an image.

Positive image A visual feature which is indicative for a desired object class, i.e. which appears on instances of the desired object class and not regularly in background image areas .

Positive pixel A pixel belonging to an instance of the desired object class.

Positive image An image contaning at least one object instance.

Weak label A label denoting whether an image is a positive image or a negative image. It is "weak" since it is less descriptive than a bounding box.

1. INTRODUCTION

Today, the number of publicly available digital images is vast due to the growth of social media websites and the omnipresence of digital cameras. On the one hand, researchers in the field of digital image analysis are hence provided with virtually unlimited data. On the other hand, this data is usually not readily usable for training any retrieval or classification approaches since it does not come with annotations. While images on the Internet are sometimes assigned *global image tags*, i.e., keywords indicating the content of the respective image, more detailed annotations such as bounding boxes for objects of interest are rare.

More precisely, finding images showing a certain object is usually not difficult, since many online photo sharing platforms such as Flickr[1] provide search engines which allow retrieving images for given query tags. Yet determining the location of the respective object within these images, e.g. by a bounding box, must usually be done manually - a time-consuming and tedious task and sometimes even impossible. Such more detailed annotations are, however, often required for training state-of-the-art image classifiers or building databases for image retrieval. Therefore automatically determining such annotations is a problem worth examining.

1.1 Problem Definition

In this thesis a method for automatically determining object annotations in the form of bounding boxes from weakly labeled data is discussed. Another concern of this work is discussing the main aspects of the problem of automatic annotations. We hence first explain the terms of this problem in our context and define our task in detail.

We begin by defining an *object class* as a category which subsumes real-world objects which can all be semantically assigned to this category. In

[1] http://www.flickr.com/

other words, the name of the object class can be used to refer to each object belonging to this class. Our approach thereby aims at object classes which possess relatively low intra-class variance with regard to visual features. We hence consider specific object classes such as the brand logo of one specific brand. We explicitly do not consider more abstract object classes like "building" or "vehicle" which are usually characterized by a large intra-class variance since they subsume multiple sub-classes. Also, we only aim at annotating objects from one class at a time, while we explicitly do not assume that the desired object always appears alone in images, i.e., no other objects may appear alongside the desired object.

The object annotations we want to determine are bounding boxes around object instances in positive images. A *positive image* is thereby an image which contains one or multiple instances of a desired object class. As already mentioned before, a *bounding box* is thus a rectangle within a positive image which tightly encloses an object instance.

By "weakly labeled data" we denote sets of images with binary, global labels. In this context "binary" means that an image may either be positive or negative and "global" indicates that we only have one single label for a full image. Thus, we assume that we are provided a set of images for which we know that they are positive images as defined above, and a set of negative images for which we know that they do not contain any images of the desired object except for negligible noise.

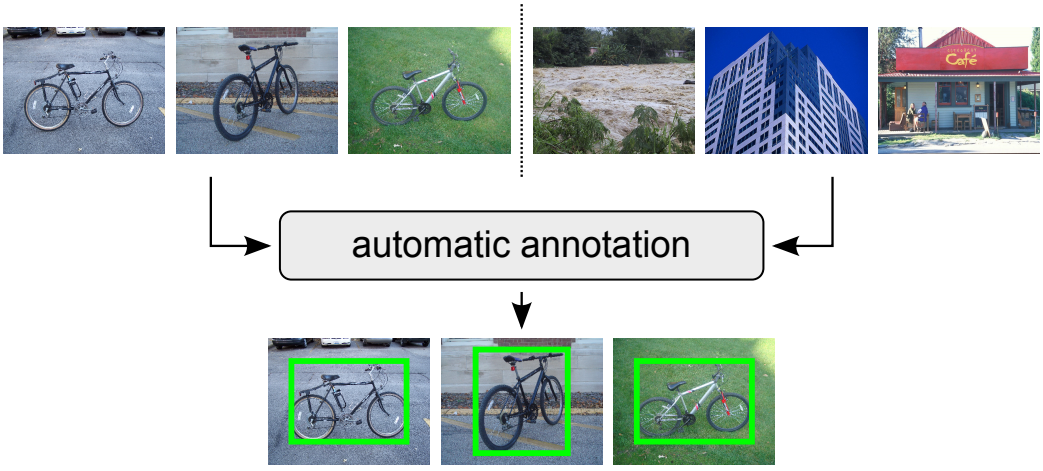


Fig. 1.1: Visualization of the task of automatic object annotation.

Figure 1.1 visualizes our problem definition. Altogether, the input to our

method are two sets of images consisting of positive and negative images, respectively. The output is then a set of one or multiple rectangles per positive image which are supposed to enclose the object instances present in the respective image.

1.2 Assumptions

Our method requires that a number of assumptions about the image sets hold.

As mentioned above, the desired object must feature little intra-class variance with regard to the visual features we use for our approach, namely color and gradient features. If the desired object's appearance is not relatively consistent across the positive images, it cannot be distinguished from random background by our method. A related requirement is that the background of the positive images is more diverse than the wanted objects themselves with regard to the visual features we examine. If the positive images all share background regions which are visually almost identical, we cannot decide that this region is not part of what we are searching for. Even though these are relatively constrictive requirements, they are still realistic for a large number of object classes. Also, if we do not assume any knowledge beyond global image labels, these requirements are not far-fetched.

For the negative set, we also have two requirements: First, the negative set must be to some degree representative for the background of the positive images. This does, however, not mean that our negative set must consist of the exact same scenes constituting the background areas of the positive images. The negative images must in fact contain a large variety of background images such that it is likely that visual properties from the background of the positive images are also encountered in negative images. In practice, we can thus usually use random photos for our negative set. Another obvious requirement is that the negative set may not include images containing the desired object except a very small number of mistakes which are called *noisy images*.

1.3 Overview

Our approach to automatic object annotation is divided into two major steps which both estimate bounding boxes for a set of positive images. We first

apply an initial statistical model on visual features and then use a latent structural Support Vector Machine (SVM) training algorithm for improving the bounding boxes determined by the initial model.

The initial model is based on the observation that, given our assumptions mentioned above, visual features which appear more frequently in the positive images than in the negative images are candidates for features which are indicative for the desired object. For these candidates we introduce a confidence-based decision function which discriminates such features from features which are likely to be background features.

We use two implementations of the initial model based on two different feature types. The first feature type are pixel colors for which we experimentally determine a suitable set of parameters and explain an enhancement which allows us to use a histogram-based color model instead of directly applying the initial statistical model. The second feature are clustered Histogram of Oriented Gradients (HOG) features. Both features yield a set of positive pixels for each positive image which we merge by intersecting both sets. Afterwards we use a heuristic to estimate one or multiple bounding boxes based on the resulting pixel set. The heuristic is based on fitting Gaussian mixture distributions into the positive pixels.

The resulting bounding boxes are then used as input for a latent structural SVM training algorithm which is an implementation of the Convex-Concave Procedure (CCCP) algorithm. The CCCP algorithm simultaneously learns an SVM model and determines new estimations for bounding boxes since we treat the bounding box positions as latent variables. Upon termination, the CCCP algorithm returns a final estimation for the bounding boxes.

All our methods are evaluated on three publicly available datasets by means of overlap-recall statistics. The color model is also evaluated on an additional dataset for the special problem of skin detection. Our final experiments also evaluate our bounding box estimations in a retrieval scenario and in an image classification scenario.

The remainder of this thesis is organized as follows. In the next section, we provide a listing of previous and related work. Chapter 3 introduces the datasets used throughout this thesis. In chapters 4 and 5 we describe our initial statistical model and how we use it do determine bounding boxes. The following chapters 6, 7, and 8 describe our implementations of the initial model for colors, gradients, and the combination of both, respectively. Afterwards, chapter 9 describes our heuristic for estimating multiple bounding boxes based on positive pixels provided by the combined initial model.

Chapter 10 then explains and discussed the latent structured learning algorithm in detail. The final bounding boxes are then evaluated for applications in chapter 11. The final chapter 12 summarizes and concludes the thesis.

2. RELATED WORK

In this chapter we list previous works on which this thesis is based and works which deal with a similar problem.

2.1 Previous Work

This thesis subsumes previous work described in [31] and [33]. In [31], a method for creating a discriminative color model from global image labels is presented. The implementation of our initial statistical model for color features is based on this method. The follow-up publication [33] enhances [31] by adding HOG features and formulating the goal of automatic annotation. It also introduces an SVM-based training algorithm for improving bounding boxes. In many regards, the thesis at hand is thus based on [33]. However, in [33] the discriminative feature models for color and HOG require different parameters while in this thesis a unified model for both features is introduced. Also, the custom SVM algorithm of [33] is replaced by a latent structural SVM training algorithm. Besides, in [33] it is assumed that the rough aspect ratio of the wanted object is known in advance and only one bounding box is estimated per image while in this work we consider multiple instances per image and estimate aspect ratios automatically.

The color model described in detail in section 6 is divided into two stages whereas the second stage (explained in section 6.2) borrows from the work of Jones and Rehg [23]. We use the same histogram-based approach for creating a discriminative model from positive image regions. The main difference is that our positive regions are not manually annotated but deduced from color statistics and a flood fill algorithm.

In chapter 7 we describe how we implement our statistical model for gradient-based features, namely Histograms of Oriented Gradients (HOG). The HOG features were first introduced by Dalal and Triggs [12]. Our implementation of the HOG features is based on the variant suggested by Felzenszwalb et al. [20].

For the initial model as well as for our CCCP algorithm and the final evaluation in an image classification setting, we use the Bag-of-Visual-Words paradigm proposed by Sivic and Zisserman [42] and for instance also explained in [18].

In the second stage of our method, we use a training algorithm which solves a structured latent learning problem. The idea of using support vector machines for structured output spaces was introduced by Tsochantaridis et al. [45]. A latent enhancement of structural SVM was suggested by Yu et al. [50, 51].

The CCCP algorithm we use for improving our bounding box estimations was proposed by Yuille and Rangarian [52]. As optimization algorithm for the CCCP algorithm, we use the cutting plane method as proposed by Joachims [22]. A number of further implementation decisions for the SVM training algorithm were inspired by the work of [53] who used latent structural SVM for object detection. Our implementation of the latent structured learning algorithm is also based on [51].

In our final section, we perform image retrieval using a method introduced by Romberg and Lienhart [35]. One motivational problem for this thesis is visual adult image recognition [32] which is a task where automatic annotation is exceptionally desirable.

2.2 Work on Related Problems

The idea of learning from weakly labeled data or determining training examples in unlabeled or weakly labeled data has been the topic of many recent works. In this section, we name a few selected examples, where this list does not claim to be exhaustive.

In the work by Tang et al. [43] discriminative segment annotations from weakly labeled video are determined. More specifically, the input to their approach is a video labeled with a weak verbal tag. For this video, spatio-temporal segmentation is performed resulting in a number of candidate regions which can be identified consistently across multiple video frames. In a scenario called "transductive segment annotation", the authors then use segments from positive and negative video frames for determining if the candidate regions are positive.

In another recent work by Chen et al. [10], synthetic training examples in weakly labeled videos are searched using only a few manually labeled training

examples. While their scenario and assumption are quite different from our task, their problem is still similar to automatically annotating objects, since they deduce a model for positive instances from only a few known positive examples.

Another interesting approach with a similar goal as automatic annotations is suggested by Alexe et al. [2]. Here, the authors define an "Objectness" measure, which is capable of determining whether any given bounding box within an image is likely to contain an object. This method obviously needs a distinct definition of the term "object" which is discussed in [1] by the same authors. The objectness measure has a similar goal as our method since it tries to find regions of interest without actually assuming any further information about the actual object class. Still, it requires a set of training images where objects are annotated manually.

Also, the concept of Multiple Instance Learning [14] (MIL), as for instance used in [26, 48], is related to our problem. Transferred to our context, MIL deals with the problem of learning an object model from sets of instances which are labeled as positive or negative as opposed to individually labeled training instances. Usually, in MIL problems one assumes that the positive set may contain negative images as opposed to our assumption of a positive set with only positive images. Still, a similar statistical estimation is performed as for our initial model.

Works with similar goals as ours, often in some way exploit the fact that large negative data is easily obtainable which is also an assumption of our method. Another interesting work which should hence be mentioned in the context of our problem is [41], where the usefulness of large amounts of negative data is discussed.

We also want to mention the work of Barnard et al. [3] where images are labeled automatically with verbal tags. They use the term "auto-annotation" for the task of assigning global tags to (full) images, i.e., using the terms we use in our our work, they automatically assign global labels to images. Also, they automatically assign tags to image segments which is similar to our task of finding bounding boxes. However, in contrast to our work, the task addressed in [3] is not finding the common object in a set of images but assigning known verbal labels to images and image segments automatically. In other words, a joint distribution between image segments and words is learned. Note that, since the segments they annotate include scenery classes like "sea" and "sky", the features they use include sizes and positions of image segments, which we explicitly consider not suitable for the object classes we

examine. Further features used in [3] are color and texture, which is similar to our set of features.

An approach by Blei and Jordan [9] deals with the same tasks as [3]. Labels are assigned to images and image regions based on a probability model derived from known correspondences between images and verbal tags. It uses a similar set of visual features as [3] and derives the verbal tags from given image captions.

Felzenszwalb et al. [20] suggest a successful approach to object detection with discriminative part-based models. Their work is interesting in the context of our work for two reasons. First, the task of object detection is naturally related to the task of automatic annotation, and [20] is among the state-of-the-art approaches to this problem. Besides, we use a similar HOG implementation as [20]. Second, in [20] properties of the ground truth are modeled by latent variables, which is similar to the second stage of our approach. Specifically, the view point annotation of the training data is discarded and replaced by a mixture model with latent correspondences between training examples and mixture models. Furthermore, the relative part locations of the part-based object models are latent. Also, to a certain extent the problem of non-optimal bounding box annotations is addressed, since the location of the desired objects in training images is considered a mutable property.

In [15], Santosh et al. discuss another problem which is similar to automatic annotation. They try to find consistent sub bounding boxes based on given bounding boxes. Specifically, their approach assumes that object bounding boxes are given and by aligning these bounding boxes, sub rectangles are detected which appear across all given examples. They also use a latent SVM approach based on HOG features. Note that this approach is also similar to the aforementioned approach by Felzenszwalb et al. [20] who also model object parts with latent positions.

Santosh et al. already postulate in their work that single bounding box annotations for objects are less expressive than multiple bounding boxes per object. In other words, forms of annotations exist, which do not require labeling on pixel-level but still yield a better object description than bounding boxes. Another main disadvantage of rectangular bounding boxes is that objects are often non-rectangular and thus bounding boxes include background regions. Monroy and Ommer [27] address this problem by using Multiple Instance Learning for automatically segmenting annotated objects by removing cells from their HOG feature grids which are likely to be in the background.

Similarly to the former approaches, the work by Schnitzspan et al. [38] also aims to automatically overcome bounding box drawbacks by finding meaningful object parts within given bounding box annotations. Thereby consistent sub regions are detected and background is removed. Also, the authors emphasize the problem of different object articulations, i.e., different poses and view points as a disadvantage of single bounding box annotations. Again, densely computed HOG features are used and a latent learning algorithm is applied. However, in [38] conditional random fields are used instead of Support Vector Machines.

In another work dealing with an analogous problem as this thesis, Blaschko et al. [8] learn object models from partially weakly labeled data in the scenario of people detection. In contrast to our approach, a certain amount of training images are assumed to come with bounding box annotations. For a significant proportion of their training data, they assume that the actual bounding box annotations are unknown and model them as latent properties. The training algorithm is then also based on the CCCP algorithm and similar to our approach, useful negative examples are determined during training by searching for most violated constraints. The features used are again HOG features which are effective for the problem of people detection in case of relatively low pose variance.

3. DATASETS

In this thesis four different datasets are used which have different properties accounting for various experimental observations. The datasets are introduced in the following sections.

3.1 Brand Logos

The first dataset we use is the FlickrLogos-32 [36] dataset. It consists of 32 sets of images containing instances of 32 different brand logos. Each set contains 70 images showing one or multiple instances of the logo of the same brand. However, almost every image also features background. In some images the brand logo is not even featured very prominently.

We choose the FlickrLogos-32 dataset, since brand logos are rigid objects for which our main requirements explained in detail in section 1.2 hold. A brand logo usually has very distinct visual features and background regions are unlikely to be visually similar to object instances. Also, the intra-class variance is usually very small for brand logos.

Another advantage of this dataset is that both feature types we use, i.e., color and gradients, are naturally suitable for many brand logos. Figure 3.1 shows a few examples images from six logo classes from FlickrLogos-32: "DHL", "Coca Cola", "Esso", "Aldi", "Pepsi", and "Shell". These six classes are fairly representative of the whole dataset with regards to difficulty and are used throughout this thesis as example logos which are evaluated individually in our experiments. For the remainder of this work, we refer to this sub set as FlickrLogos-6.

FlickrLogos-32 comes with pixel-wise annotations as well as bounding box annotations for each logo instance.

Fig. 3.1: A few example images for the six classes of FlickrLogos-6 datasset (a subset of FlickrLogos-32).

3.2 Flowers

The second dataset we use is Oxford 17 Flowers [29]. It consists of 17 different classes of flowers with 80 images each. For some images, pixel-wise annotations are provided in the form of color maps which indicate actual positive pixels. Note that not all images are annotated which means that

our evaluation only considers a subset of 848 images. Also, not all classes are represented by the same number of annotations - one class even has no annotations at all.

For the images which have annotations, we deduce bounding box annotations from the pixel-wise annotations by determining minimal bounding rectangles for the annotated positive areas. Note that this leads to incorrect bounding boxes if multiple instances overlap. We therefore re-work obviously incorrect examples by manually separating multiple objects in the annotation images. Still, for a few instances evaluation is noisy, especially for classes where it is ambiguous if a patch of many small blossoms constitutes one large or many small instances. Also there is a small number of false positive annotations due to small sub regions which are separated from the actual object instance.

Fig. 3.2: Example images for three classes from Oxford 17 Flowers.

For most classes, the flowers dataset also fulfills our requirements of relatively low intra-class variance and object distinctness with regards to image background. Note however, that different from brand logos, flowers do

not have strong gradient features considering the difference to the respective backgrounds. Also, the background regions of flower images are far less variable than brand logo backgrounds. Therefore, in this regard, Oxford 17 Flowers is less suitable for our method and thus more challenging than the logo dataset. Besides, some flower classes feature instances with a large number of different colors or flowers which do have non-discriminative colors such as white.

Figure 3.2 shows a few example images from three different classes. Note that the second row shows one of the aforementioned classes with large color variance while the other classes (as most flower classes) have a relatively fixed color scheme.

3.3 3D Object Categories

The 3D Object Categories [37] dataset consists of images from ten different object classes: bicycle, car, cellphone, head, iron, monitor, mouse, shoe, stapler, and toaster. For each object class, the dataset features ten different object instances whereas for each instance, 72 images are provided which vary in view point. The view points are annotated and defined by distinct distance (i.e., scale of the object within the image), azimuth, and altitude. Also, several different backgrounds were used for each object instance.

Since extreme view point differences result in high visual differences for most objects, we do not use all available view points. For instance consider bicycles where the front view can arguably be considered a different class then the side view with regards to visual properties. We hence only use a sub set of view points whereas we allow high variation in distance but limit the variation of azimuth and altitude. In detail, we only use objects annotated as "view from right side" with the two neighboring azimuths ("rear-right view" and "front-right view") and for the "view from right side" we allow three different altitudes. We allow all three given viewing distances for each of the five selected viewing angles for a total of 15 view points per object instance. The dataset consists of ten instances per class. Since not all classes provide each of the selected views for each object instance, we obtain a total of 1430 images.

Note that the ten different instances from each object class may highly vary in appearance. Thus, considering the different view points used, we overall still have a reasonable amount of intra-class variance.

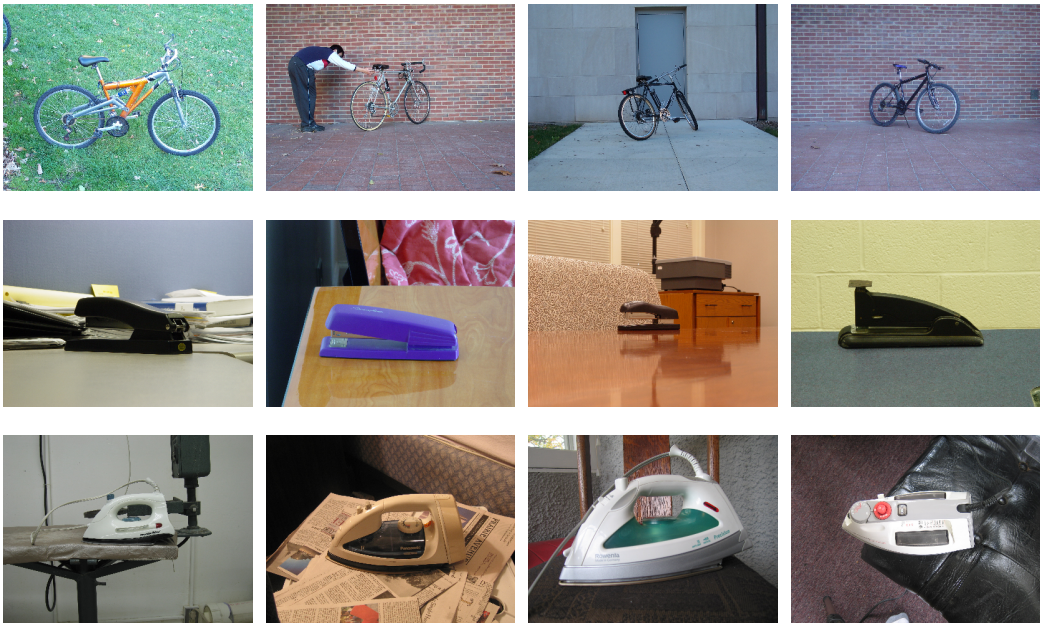


Fig. 3.3: Example images for three classes from 3D Object Categories: "bicycle", "stapler", and "iron".

In contrast to the Oxford 17 Flowers dataset, from the 3D Object Categories dataset we expect a higher intra-class variance with regards to color but (in comparison to flowers) a lower intra-class variance with regards to gradients. Thus, for this dataset, the gradient feature can be expected to be more suitable than for flowers. Figure 3.3 shows a few example images.

3.4 Bikini

Our final dataset is the Bikini dataset which features images of people in swim wear. The dataset consists of 12,500 publicly available images which were downloaded from photo sharing website Flickr by using the query tag "bikini". The images were manually selected such that each image actually shows at least one person. For a subset of 2015 images, bounding box annotations were created manually. Even though there is a noticeable bias towards beach or pool scenes, the backgrounds of the Bikini images are still more diverse than in the flower images.

This dataset is only used for evaluating the color model we introduce,

since color is apparently a suitable feature for people in swim wear. Gradient-based features, however, are usually subject to large variations among people (if no pose constraints are imposed) and while, under certain assumptions, HOG features work on upright pedestrians [12], they are not suited for photos where people may have arbitrary poses and may be arbitrarily truncated.

Figure 3.4 shows a few examples of the bikini dataset.

Fig. 3.4: Example images from the Bikini dataset.

3.5 Negative Sets

For most of the tasks discussed in this thesis, a relatively large negative image set is required, i.e., a set of images which does not show instances of the desired object of the respective dataset. Since the aforementioned datasets only consist of 70 to 150 images per class, we simply use the negative set of FlickrLogos-32 as negative set (exceptions are explained in the respective sections below). The negative set of FlickrLogos-32 is a set of $6,000$ random photos downloaded from Flickr using various tags. The tags cover a wide range of different topics and due to the high rate of incorrectly or imprecisely labeled images on Flickr, we consider our negative set a set of random photographs. It is thus a reasonable choice as a background dataset for many classes, since it contains a large number of potentially realistic backgrounds.

For the Bikini dataset and for some of our experiments,

we also use another negative set of $100,000$ images, called World 100k, which was acquired in the same way as the negative set of FlickrLogos-32, so both datasets are very similar. Note that none of our negative sets have been manually filtered except for the negative set of FlickrLogos-32 which does not contain any logos. All other object classes may be present in the negative sets, but only in a few images.

Figure 3.5 shows a few example negative images (from World 100k).

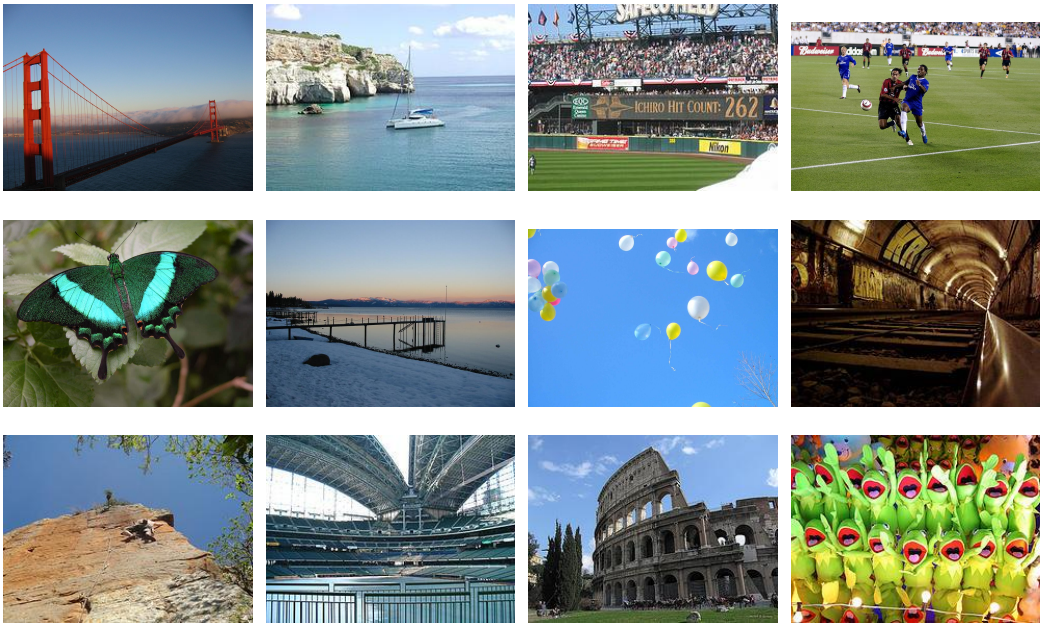


Fig. 3.5: Example images from World 100k.

4. DISCRIMINATIVE FEATURE MODEL

In this chapter we explain how we create a feature model for a given object class. It defines the features which are likely to identify instances of the desired object class. Thus our task is to determine a suitable object-related subset of a given finite set of features.

Let F be a finite, discrete set of visual features f which can be found in images. For example f may be a certain color from a discrete color space or a gradient-based feature. We now want to create a discriminative model $c(f) : F \rightarrow \{0, 1\}$ which assigns a boolean value to each feature $f \in F$. The boolean value indicates whether f is a *positive* feature (i.e., $c(f) = 1$) or a *negative* feature (i.e., $c(f) = 0$). A positive feature is a feature which is indicative for our wanted object class, e.g. a color which is typical for the object, while a negative feature is not. Note that being indicative for the wanted object not only means that feature f is regularly found on the wanted object, but also that feature f is at the same time less common for background areas. This is important since we explicitly do not search for features which are typical for the wanted object class if they are also very common in background images. In other words, if a feature which is typical for the wanted object class, is also common in random background, we will not consider it indicative or a positive feature. Based on the positive features we find in an image, we can determine positive pixel locations and finally regions of interest.

Recall that we do not know the locations of the wanted object instances within our positive images. We thus cannot observe directly which features within the positive images belong to the wanted object and which features describe the background. The only information we exploit is a binary global image label, i.e., we have a set P of positive images which are guaranteed to show the wanted object at least once. The locations of the object instances are unknown. Also, we are given a larger set N of negative images which does not contain instances of the wanted object except for a negligible number of noisy images.

In this context, we now restate the assumptions about our dataset from section 1.2 more specifically: Since we do not have further information, we have to assume that two major assumptions about the image sets P and N hold if we want to deduce which features are positive, i.e., indicative for the desired object:

1. Objects of the desired object class occur in a limited number of different appearances with regards to the observed features F. In other words, there exists a subset of features $F_{pos} \subset F$ that is indicative for the object class.

2. Background areas of positive images must be similar to the negative images and with respect to the observed features more diverse than the wanted objects. Put differently, it is a property of the background that no features are indicative for it.

The first assumption is intuitive for any discriminative model, since if the object instances do not share a reasonable number of indicative features and hence each instance has its own individual set of features, the respective class cannot be represented by a discriminative model based on the respective feature type. For example, (arbitrary) cars cannot be modeled very well by a discriminative color model, since theoretically cars can appear in every possible color. The second assumption requires that the image regions showing the wanted object must have more in common across all positive images than the respective background areas. This assumption must hold since we do not know the location of the objects within the positive images and thus we can only search for features which the positive images have in common. If, however, the positive images often share the same background features, which in the worst case do not appear very often in actual background images, we cannot identify these features as non-object features. Note that FlickrLogos-32, Oxford 17 Flowers, and 3D Object Categories fulfill these requirements to certain degrees.

Furthermore, the negative set must be reasonably large, since we need it to estimate a representative background model. A useful background model can only be derived from a set which contains sufficiently many different scenes and concepts such that features which are not indicative of the desired object class are present in the negative set in representative quantities. As the Internet provides a huge number of publicly available images with large variety, and our model is tolerant to some noise, it is not difficult to provide

large negative (background) sets for many object classes. In most cases, the negative set of FLickrLogos-32 (which consists of random photos) can thus be used.

Note that the lack of object annotations means that we cannot actually learn an object model or train a standard object detection classifier for two reasons. First, we do not have object examples and thus no positive features for training a standard object detection classifier. Second, we cannot evaluate a classifier or model, i.e., we cannot decide if an estimation for an object location is correct. For these reasons, we have to use empirical heuristics and statistical models as explained in the following sections.

4.1 Statistical Model

Our task is to provide a statistical model which determines a subset F_{pos} of all features F which can be used to identify the wanted object. Thus, we need features which appear on the wanted object significantly more often than on random background. The main problem is that we do not have any object annotations and thus do not know how large the wanted objects are in each image (i.e., how many features they produce) or whether a feature observed in a positive image appeared on the wanted object or in the background. Therefore, we cannot compute any actual relative frequencies in order to obtain a probability $P(f|object)$ of observing feature f on an object.

An observation which is independent of the actual object size, however, is the number of positive images which contain feature f. The only obvious constraint on the object size is then that the object must not be too small to produce any features of the given feature type.

In other words, we know the probability $P_P(f(I) = 1)$ of randomly drawing an image I from set P which contains feature f. For brevity, we let $f(I) : I \rightarrow \{0, 1\}$ denote whether image I contains feature f at least once wheres $f(I) = 1$ denotes the event that I does contain f. For instance, if f is a color $f(I)$ means that at least one pixel of image I has color f. We do not require a feature to appear more often than once since we do not want to introduce a threshold which may not be suited for all feature types. For example, we will use visual words as features which often appear only once within an image, in particular if they belong to a given object class. Then, $f(I) = 1$.

We of course still do not know if the presence of feature f in image I was

caused by the wanted object or by the background. In fact, $P_P(f(I) = 1)$ consists of two components which represent the different possible feature appearances. First, we have to consider the relative number $P'_P(f(I) = 1)$ of images in which f is present on the wanted object and not in the background. The second component of $P_P(f(I) = 1)$ is the relative number of images $P''_P(f(I) = 1)$ in which the feature was in the background and potentially at the same time also on the wanted object. Together, $P'_P(f(I) = 1)$ and $P''_P(f(I) = 1)$ yield the total number of images containing feature f:

$$P_P(f(I) = 1) = P'_P(f(I) = 1) + P''_P(f(I) = 1). \tag{4.1}$$

The quantity we are interested in is the number of images in which the object is present only due to the presence of the wanted object, i.e., $P'_P(f(I) = 1)$ which we cannot observe directly due to the lack of annotations. As stated above, we can, however, observe $P_P(f(I) = 1)$ since it is simply the relative number of images which contain feature f regardless of whether the feature occurred on the object or in the background. Note that $P_P(f(I) = 1)$ is obviously an upper bound to $P'_P(f(I) = 1)$, i.e.,

$$P_P(f(I) = 1) \geq P'_P(f(I) = 1). \tag{4.2}$$

Recall that we have defined positive features as features which are regularly present on the wanted object but usually not in the background at the same time. Therefore, for positive features characterizing the wanted object, we can expect $P''_P(f(I) = 1)$ to be close to 0. As a consequence, $P_P(f(I) = 1)$ is usually a fairly accurate estimation of $P'_P(f(I) = 1)$ if the feature is indicative for the wanted object.

Thus, we now assume $P'_P(f(I) = 1) = P_P(f(I) = 1)$, which means that we can only overestimate the actual probability $P'_P(f(I) = 1)$. However, by this assumption we disregard any occurrences of feature f in the background of the positive images (i.e., we implicitly assume $P''_P(f(I) = 1) = 0$). So for background features this estimation may be misleading. Also, if $P_P(f(I) = 1)$ is relatively small, f may still be an object feature if it is usually not observed in random background images. For these reasons, we need to also consider our negative set N.

For the negative set N, let $P_N(f(I) = 1)$ be the relative frequency of negative images which contain feature f. We know that in N each occurrence of a feature f was in the background, since the negative images do not show the wanted object. We allow some noise, i.e., a few negative images which

may show the wanted object, but since the negative set is supposed to be large, a few false negative images do not significantly influence the relative frequency $P_N(f(I) = 1)$. Thus, following the above notation (and ignoring noisy negative images) we know that

$$P''_N(f(I) = 1) = P_N(f(I) = 1), \tag{4.3}$$

where $P''_N(f(I) = 1)$ is the relative frequency of images in N in which feature f occurs in the background.

We therefore now have the probability $P_N(f(I) = 1)$ that a random image from a representative background set N contains feature f and a (potentially too large) estimation $P_P(f(I) = 1)$ of the probability of an image from a set of positive images containing the feature f on the wanted object. Since $P_P(f(I) = 1)$ is an upper bound of $P'_P(f(I) = 1)$ we can deduce that features for which $P_P(f(I) = 1) < P_N(f(I) = 1)$ are not indicative for the wanted object since they do not appear more often in the positive set than in the negative set.

Features for which $P_P(f(I) = 1)$ is larger than $P_N(f(I) = 1)$ are, however, candidates for positive features, i.e., features indicative for the wanted object. We now could simply compare the ratio of both probabilities to a threshold in order to determine positive features. However, since $|P|$ is usually significantly smaller than $|N|$, we define our decision function by means of a confidence analysis of the observation of $P_P(f(I) = 1)$ depending on the actual size of P. This decision function is explained in detail in the following section.

4.2 Confidence Intervals

Recall that $f(I)$ denotes a binary property of an element I of a given set, i.e., whether or not I contains at least one instance of feature f. Also, for our negative image set N we assume that it is large enough to be representative for the background of the positive images. Therefore, we regard $P_N(f(I) = 1)$ as the actual probability of a random background image containing at least one instance of feature f.

If f is a common background feature, its appearance in positive images can be explained by its appearance in negative images. In other words, a feature which is indicative for the wanted object is a feature whose appearance in positive images can not be ascribed to the fact that it regularly appears

in background images, i.e., images which do not share any specific common object. We therefore define a probability model which reflects this reasoning. Our probability model must constitute the probability that, based on our knowledge about the negative set, a feature is also likely to appear in another image set, i.e., our positive set.

For deriving our probability model, we regard our observation $P_P(f(I) = 1)$ as the outcome of sampling $|P|$ individuals from a given base set N which contains $|N|P_N(f(I) = 1)$ individuals with the binary property $f(I) = 1$. For brevity, let $n = |P|$ and $m = |N|$. Also, let $x_f := P_P(f(I) = 1)$ and $p_f := P_N(f(I) = 1)$. Statistically, the outcome of an experiment of this type is then governed by a binomial probability distribution $b_f(nx_f; n, p_f)$ over nx_f with expected value $E[x_f] = n \cdot p_f$ and variance $Var[x_f] = n \cdot p_f(1-p_f)$. Note that this distribution intuitively has the aforementioned properties: If the observed occurrence x_f of feature f can be explained by this distribution, it is likely that the feature is a background feature.

In order to determine if observed occurrence x_f can be explained by the distribution, we now analyze our confidence in observation x_f under the assumption that P is a random sample from N. We facilitate defining a confidence interval by first approximating the binomial distribution by the following normal distribution:

$$b_f(nx_f; n, p_f) \approx n^{-1} N(x_f; \mu_f, \sigma_f^2) \tag{4.4}$$

with

$$\mu_f = p_f; \quad \sigma_f^2 = p_f \cdot (1 - p_f) n^{-1}.$$

This approximation is justifiable since we usually have reasonably large positive sets (i.e., $n > 30$). Note that the variance σ^2 depends on the size of set P which reflects the fact that we expect less deviation for large positive sets. Figure 4.1 shows an example binomial distribution over nx_f for $n = 70$ and $p_f = 0.4$, and the respective approximation by a normal distribution over x_f.

Now we can define a confidence level at which we expect to observe a given value of x_f. The confidence level induces a confidence interval into which we expect the value of x_f to fall with a given probability if our assumption holds that P is a sample from N with regards to property $f(I)$. We thus define a confidence-based decision function $c(f)$ which allows $P_P(f(I) = 1)$ to deviate from $P_N(f(I) = 1)$ by a certain extent θ_f and still consider f a background feature. As mentioned above, features which occur exceptionally

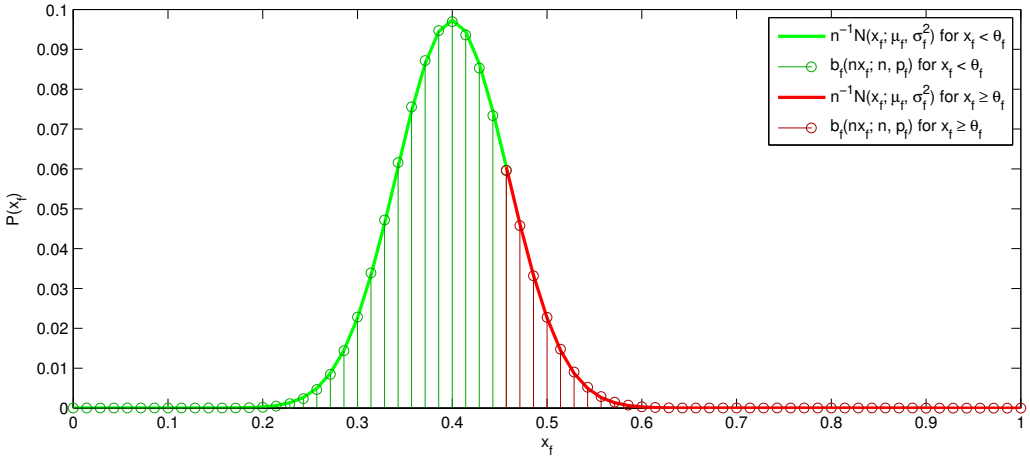


Fig. 4.1: Binomial distribution over nx_f approximated by normal distribution over x_f. Here, $\mu_f = p_f = 0.4$ and $n = 70$. The green area illustrates a one-sided confidence interval for $\theta_f = z_F\sigma_f$ with $z_F = 1$

less often in positive images than in negative images (i.e., if $P_P(f(I) = 1) < P_N(f(I) = 1)$) are intuitively no candidates for features which are indicative for the desired object. Therefore, the confidence interval becomes a one-sided interval which yields our decision function $c(f)$:

$$c(f) = \delta(x_f \notin [0, p_f + \theta_f]) \tag{4.5}$$

where $\delta(A)$ is 1 if A is *true* and 0 otherwise. Thus, $c(f)$ is 0 if f is considered a background feature and 1 otherwise. The accepted deviation θ_f from the expected value is usually defined in terms of the standard deviation:

$$\theta_f = z_F\sigma_f = z_F\sqrt{\frac{p_f(1-p_f)}{n}} \tag{4.6}$$

This definition allows determining a threshold based on the confidence level with regards to the standard deviation σ_f by a constant factor z_F. For example, $z_F = 1$ (i.e., $\theta_f = \sigma$) corresponds to a confidence level of approximately 84% for a one-sided interval.

By definition, σ_f depends on p_f such that the standard deviation is largest for $p = 0.5$ which reflects the fact that if a feature is present in half of the negative images, our confidence interval will be maximally large for a given z_F. For smaller or larger values of p_f, the confidence interval for a given

confidence level becomes smaller. Intuitively, this means that features present in very few or many negative images have smaller confidence intervals. On the other hand, the size of the interval depends on σ_f and hence on the size of the positive set n. In figure 4.1, the confidence interval is marked as the green area of the distribution for $z_F = 1$. Features falling into the red area would be considered not explainable by the background distribution and are thus candidates for positive features.

We have now reduced our problem of determining candidates for positive features to a straightforward decision function. Note that our threshold θ_f still involves a constant value z_F which determines the size of the confidence interval. The choice of this constant is explained in the following section.

4.3 Global Threshold

As explained above, we need to define the size of the confidence interval by choosing a factor z_F in equation 4.6. However, we want this value to depend on how useful a given feature type F is for the object class at hand.

In particular, if the desired object may be well discriminated from background areas in terms of feature F, we want the set of positive features F_{pos} to be correspondingly small. If, however, feature type F is less suitable for the given object class, we want F_{pos} to be a large set. The reason is that, as elaborated below, we will intersect binary masks of positive pixels stemming from different features. For intersecting binary masks, positive pixels do not affect the overall intersection result with regards to removing pixels from the intersection result. Therefore, a "weak" feature will not harm our final set of positive pixels if it produces a large set of positive features and thus a large set of positive pixels. This issue will also be explained in more detail below.

Overall, we need a factor z_F which depends on the distinctiveness of feature type F for the given object class. As a measure of this distinctiveness, we consider the quality of the one feature $f^* \in F$ which is least likely to be a background feature and thus has the minimal value of $n^{-1}N(x_f; \mu_f, \sigma_f^2)$ (see equation 4.4) among all features in F. To account for extreme outliers, we choose as f^* the feature as the median value of $n^{-1}N(x_f; \mu_f, \sigma_f^2)$ among the five features with smallest values.

Now we choose

$$z_F = \alpha z_F^* \tag{4.7}$$

where z_F^* is the factor for which only f^* is included in F_{pos} and α is an empirically chosen constant. Now our confidence interval directly depends on the median of the five "best features". Still we have an empirical constant α, so at first glance we have just traded in one constant (z_F) for another (α). However, α is a more intuitive factor since it directly describes the relative distance to feature f^* instead of a confidence level as z_F. Also, if we directly choose z_F as a fixed value (independent of f^*) it would not fulfill the requirement that "bad" features lead to large sets F_{pos}. Besides, for some features we might then obtain empty or extremely large sets F_{pos}.

For illustration, consider the distribution in figure 4.1 where $z_F = 1$. The confidence interval only excludes relatively few features from F_{pos}; almost every feature which appears more often in positive images than in negative images will be considered positive. The shown confidence interval is thus desirable for weak features, for instance color features on objects with large intra-class variance with regards to color. If z_F is fixed at $z_F = 1$, however, a stronger feature type will also produce a large set F_{pos} as the confidence interval will remain the same as depicted. This is not desirable since for features which have a strong "best feature", we do not want to include many additional (and potentially surplus) features. Since according to equation 4.7 z_F depends on z_F^*, we will yet obtain a much larger confidence interval and thus a small set F_{pos}. In figure 4.1, a strong feature f^* would hence enlarge the confidence interval. We experimentally determine $\alpha = 0.75$ as a reasonable value. Intuitively, z_F^* defines the distance of the reference feature's occurrence x_f^* from expected value p_f in terms of standard deviations (cf. equation 4.6). Features become hence positive if they have at least 0.75 times the distance (in terms of standard deviations) of the reference feature from the expected value p_f.

It should be mentioned that in some cases we may obtain positive features which appear in a very low number of positive images (if they appear in an even lower number of negative images) and are thus unlikely to be actually discriminative for the object class. However, since these features only appear in a low number of positive images, we do not introduce another threshold for filtering out such features.

Finally, given θ_f the positive features F_{pos} are determined by our decision function $c(f)$ as defined in equation 4.5:

$$F_{pos} = \{f \in F | c(f) = 1\} \tag{4.8}$$

Fig. 4.2: Examples for positive features found for class "DHL" from FlickrLogos-6. A few training images are shown on the left. In the center all positive colors are visualized. On the right, three positive HOG features are illustrated by a few selected image patches.

For illustration, we show two example sets of positive features in figure 4.2. We visualize positive features $F_{c,pos}$ based on a color feature and positive features $F_{hog,pos}$ based on clustered gradient features (HOG) features. These feature types are explained in chapters 6 and 7. For now we only want to illustrate two positive feature sets. On the left of the figure, four training images of the class "DHL" from FlickrLogos-6 are shown. The center of the figure shows all positive colors determined by our statistical model. On the right, patches belonging to positive clustered HOG features are shown.

In the next chapter we explain how we determine regions of interest based on the binary decision function defined on features. In the subsequent chapters we then describe implementations of our discriminative model for two different feature types, namely color and local gradient features.

5. REGIONS OF INTEREST BASED ON DISCRIMINATIVE MODEL

A standard means of describing object locations often used for manual annotation are bounding boxes. As mentioned before, a *bounding box* is a rectangular region within a positive image which tightly encloses the respective object. Hence a bounding box is a rectangle $r = (x, y, w, h)$ with upper left corner $(x, y)^T$, width w, and height h, specified in image coordinates.

Bounding boxes usually include some background area, since objects are often not rectangular. However, bounding boxes have a number of advantages. First, they can be parameterized by only four values (as opposed to annotations on pixel-level). Second, visual features often are computed on a dense grid so rectangles are a natural way to define sub regions and hypotheses for search algorithms. Besides, bounding boxes are often used for manual annotation since they are relatively easy to draw by hand.

5.1 Gaussian Bounding Boxes

In this section, we explain how we create Gaussian bounding box estimations based on positive features and their locations within positive images. We therefore first explain how we deduce positive pixels from positive features and then describe how we fit a rectangle into the positive pixel distribution.

5.1.1 Binary Maps

We need to deduce bounding boxes from our positive features F_{pos} defined in equation 4.8. Since we only use local features, each feature $f \in F$ is associated with one or multiple pixel locations $\mathbf{p} = (x_p, y_p)^T$ in a given image. Note that feature f may appear at several locations in the image and may also cover patches of multiple pixels. Thus multiple features may cover the same pixel locations and vice versa.

As a consequence, we must determine all features $F|_{\mathbf{p}}$ which are associated with a given pixel location $\mathbf{p}$. For instance, $F|_{\mathbf{p}}$ may simply contain the color of pixel $\mathbf{p}$ or all local feature descriptors which cover image patches including $\mathbf{p}$. Thus, the association between a feature and a given pixel depends on the feature type and is hence explained in the respective sections on implementations below.

If at least one of the features associated with pixel $\mathbf{p}$ is positive, i.e., if $F|_{\mathbf{p}} \cap F_{pos} \neq \emptyset$, we say that $\mathbf{p}$ is a *positive pixel* with regard to feature set F. The positive pixels of a given image i now form a set $B(i)$:

$$B(i) = \{\mathbf{p} \in P_i | F_{pos} \cap F|_{\mathbf{p}} \neq \emptyset\} \tag{5.1}$$

where P_i are all pixel locations of image i. Note that $B(i)$ may be interpreted as a binary map of the image since it simply defines for each pixel of image i whether it is a positive pixel.

5.1.2 Bounding Boxes on Positive Pixels

We now define bounding boxes based on two-dimensional Gaussian models which we fit into the two-dimensional distribution of positive pixels in $B(i)$ of image i. Intuitively, we implicitly assume that the two-dimensional positive pixel distribution has been generated by a Gaussian distribution or mixture (i.e., the weighted sum) of k Gaussian distributions. Here the underlying assumption is that multiple positive objects form multiple "dense blobs" of positive pixels which can be roughly explained by mixed normal distributions.

Fitting a Gaussian mixture model into the pixel distribution is equivalent of finding k 2D normal distributions $N_j(\mu_{i,j}, \sigma_{i,j})$ with $j \in \{1...k\}$ which best explain the underlying spatial distribution of $B(i)$. One standard way for determining $\mu_{i,j}$ and $\sigma_{i,j}$ for each j is the *Expectation Maximization* algorithm as for instance described in [5].

After fitting k normal distributions to the positive pixel distribution, we obtain k bounding boxes $\hat{r}_{i,j} = (x_{i,j}, y_{i,j}, w_{i,j}, h_{i,j})$ as follows:

$$\begin{aligned} x_{i,j} &= \mu_{i,j,x} - \beta\sigma_{i,j,x} \\ y_{i,j} &= \mu_{i,j,y} - \beta\sigma_{i,j,y} \\ w_{i,j} &= 2\beta\sigma_{i,j,x} \\ h_{i,j} &= 2\beta\sigma_{i,j,y} \end{aligned} \tag{5.2}$$

The factor β is used to enlarge the bounding box, since for a normal distribution, the interval $[\mu-\sigma, \mu+\sigma]$ only covers 68.2% of the underlying density.

According to a parameter sweep in the experimental setup explained below on our largest dataset (Bikini images), a factor of $\beta = 1.6$ was identified a good choice which we use for all experiments.

The choice of the number k of distributions for the case where we want more than one object per image is explained in chapter 9 on multi-object detection. For the preliminary experiments of the following chapters, we always assume $k = 1$ for convenience.

5.2 Quality of Bounding Boxes

Let $\hat{r}_{i,j}$ be a bounding box estimation in image i as explained above. Now let $r_{i,l}$ be a ground truth rectangle in image i from our positive image set P. We then define the *overlap* between $\hat{r}_{i,k}$ and $r_{i,l}$ by

$$o(\hat{r}_{i,j}, r_{i,l}) = \frac{\hat{r}_{i,j} \cap r_{i,l}}{\hat{r}_{i,j} \cup r_{i,l}} \tag{5.3}$$

This overlap definition is a common quality measure for bounding boxes and was for instance used in the Pascal VOC Challenge [17]. If an image shows multiple object instances, $r_{i,l}$ is considered to be the one instance with which our estimation produces the best overlap whereas each ground truth rectangle can only be assigned to a single estimated rectangle. All estimations overlapping with $r_{i,l}$ except the one with the maximum overlap are considered false positive detections unless they overlap with another ground truth rectangle for which the same conditions apply.

We want to stress that the ground truth $r_{i,l}$ is only used for evaluation and not assumed to be available for training.

It should also be emphasized that this quality measure comes with a few issues. First, if we find a bounding box which includes the full image, the overlap measure arguably becomes meaningless since the "quality" of this estimation depends solely on the size of the object within the image. In fact, $o(\hat{r}_{i,j}, r_{i,l})$ becomes the relative size of $r_{i,l}$ if $\hat{r}_{i,j}$ includes the full image.

Second, the overlap measure does not take the increased difficulty of finding small objects into account for two reasons. First, as mentioned above, pixels are discrete and thus when fitting a small box, the overlap quickly deteriorates with only a few pixels deviation, which is not the case for large objects. Second, the smaller the wanted object, the more background is present in the respective image and thus the probability of finding non-overlapping

rectangles increases. In other words, if the object is small, there are more bounding box hypotheses which produce an overlap of 0 than for large objects. Thus, informally speaking, finding larger objects which is usually already easier than finding small objects, is further encouraged by the overlap measure.

Fig. 5.1: A few examples illustrating the overlap criterion of estimated rectangles (green) and ground truth annotations (white). The overlap scores are given in the images below the green rectangles.

Also note that the overlap measure is relatively strict, especially for smaller objects. For illustration, figure 5.1 shows a few example detections (green rectangles) and their overlap scores. For the remainder of this work, the respective ground truth annotations are represented by white rectangles. None of the detections in figure 5.1 reaches an overlap score of 0.6

even though they arguably appear to be reasonable estimations. The lower left image shows an detection below 0.5 overlap which would be considered a false negative detection by the widely acknowledged Pascal VOC evaluation [17]. Since, however, this overlap definition can be considered a standard definition, we still use it in this work.

It should also be mentioned that if our estimated bounding box is empty (i.e., no positive pixels are present), we accordingly obtain overlap 0. However, in practice, one would simply define the full image as estimation in this case since we know the wanted object is present. For our experiments, we still maintain empty detections with overlap 0, since the overlap of a full image bounding box does not reflect the performance of our method. An empty bounding box is thus counted with overlap 0 during evaluation but replaced by the full image for later stages of our approach if applicable.

Throughout this paper we use overlap-recall curves for evaluating our different methods. An *overlap-recall curve* plots the relative number of detectable images or object instances on the horizontal axis, and the overlap (intersection divided by union) on the vertical axis in descending order. Thus, for each example in a dataset, the quality of the bounding box found for this example is given. Missed positive examples (false negatives) obtain an overlap of 0. The downside of overlap-recall-curves is that false positive detections cannot be deduced from them, so we state such numbers explicitly if applicable as explained in chapter 9.

6. COLOR MODEL

The first feature for which we implement our model are object colors. For some object classes, color is a strong indicator, for instance for human skin or for brand logos. For such classes we can deduce that certain colors must be present at the locations of the wanted objects. This chapter is based on [31].

6.1 Initial Model

We now use the discriminative model described in chapter 4 on pixel colors. A color value is usually represented by a multi-dimensional vector. For all color spaces we consider in the following experiments, we have three-dimensional color features f_c. As is common practice, we discretize each of the three dimensions of a color space into a fixed number of bins which yields a discrete set F_c of color features.

We first count the relative numbers of images containing a color feature f_c in our positive set and negative set in order to obtain $P_P(f_c(I) = 1)$ and $P_N(f_c(I) = 1)$, respectively. Then, we determine initial positive colors $F'_{c,pos}$ as explained in chapter 4 and defined by equation 4.8:

$$F'_{c,pos} = \{f_c \in F_c | c'_c(f_c) = 1\} \tag{6.1}$$

with $c'_c(f_c)$ being defined analogously to equation 4.5 on feature type F_c.

It is worth mentioning, that we may still exclude colors from our model which actually belong to the desired object if the same color appears in many background images. For example, white is a color which also appears frequently in background images and is thus considered a non-object color by our decision rule. However, we are only interested in colors which are indicative of the object and at the same time do not regularly appear on random images which do not include the object. So even if such colors appear on the desired object we still do not want to include them in the

object color model since we want to avoid overlap with common background which highly increases the false positive detection rate.

We can now decide whether a pixel $\mathbf{p}$ from the set of pixels P_i of image i is likely to belong to the wanted object. Note that the association between a color feature and a pixel is trivial since one pixel has exactly one color. Thus the definition of features $F_c|_{\mathbf{p}}$ associated with pixel $\mathbf{p}$ for creating the positive pixel set $B'_c(i)$ with regards to color for image i (according to equation 5.1) is straightforward: $F_c|_{\mathbf{p}} = \{f_c(\mathbf{p})\}$, where $f_c(\mathbf{p})$ denotes the color of pixel $\mathbf{p}$. Thus, our set of positive pixels is defined as follows:

$$B'_c(i) = \{\mathbf{p} \in P_i | f_c(\mathbf{p}) \in F'_{c,pos}\} \tag{6.2}$$

6.2 Adding Spatially and Chromatically Related Colors

So far, our initial color model $c'_c(f_c)$ is only suitable for identifying positive pixels, which are likely to be located on objects of interest. Yet the set of identified pixels is unlikely to cover entire objects. Also, if our assumption that the wanted object is more diverse than the background areas is violated, some of our top unique colors $F'_{c,pos}$ may be outliers.

Overall, the color model cannot yet be used to segment the objects of interest with a standard pixel frequency-based color model like the one described in [23]. Thus, we now create a similar pixel-based color model based on our initial model. We hence first determine positive pixels $B'_c(i)$ as described above for each positive image i. These positive pixels are used as seed pixels for a flood-fill algorithm. Starting with these seed pixels, we identify additional pixels which might belong to the wanted objects and at the same time exclude outliers.

Additional positive pixels should be neighbors of other positive pixels. We therefore first determine the set $\mathfrak{P}$ of all pixels $\mathbf{p}$ in all images of P whose color $f_c(\mathbf{p})$ is a positive color:

$$\mathfrak{P} = \bigcup_i B'_c(i) \tag{6.3}$$

Outliers are removed by applying a median filter[1] to each image's binary map defined by $B'_c(i)$. Now let $N(\mathbf{p})$ be the $4-$neighborhood of pixel $\mathbf{p}$ in its

[1] with size 13×13

respective image. We define $N'(\mathbf{p})$ as the pixels $\mathbf{p}'$ from the neighborhood of $\mathbf{p}$ which do not deviate more than a given threshold θ_{c^i} from the color of $\mathbf{p}$ in any color channel i:

$$N'(\mathbf{p}) = \{\mathbf{p}' \in N(\mathbf{p}) | \forall i : |c^i_{\mathbf{p}'} - c^i_{\mathbf{p}}| < \theta_{c^i}\} \tag{6.4}$$

where $c^i_{\mathbf{p}}$ is the value of the ith channel of color $f_c(\mathbf{p})$. We conservatively define $\theta_{c^i} = 0.02$ for all color channels i in our experiments. Note that $N'(\mathbf{p})$ only includes pixels which are both chromatically and spatially close to pixels for which we can safely assume that they are part of the wanted object. After determining the pixels of $N'(\mathbf{p})$ in each image of P we include them in $\mathfrak{P}$:

$$\mathfrak{P} = \mathfrak{P} \cup \left(\bigcup_{\mathbf{p} \in \mathfrak{P}} N'(\mathbf{p}) \right) \tag{6.5}$$

We repeat this procedure until $\mathfrak{P}$ does not grow anymore. The only difference is that in further iterations, we keep color $f_c(\mathbf{p})$ of pixel $\mathbf{p}$ from the original set $\mathfrak{P}$ in equation 6.4. In other words, we always compare the colors of new candidates to the respective original seed pixel's color. Thus, all pixels we add to $\mathfrak{P}$ may not deviate too much from colors found by our initial model. This prevents "leaking" into background across smooth object borders. As a result we obtain regions of similarly colored pixels, which we consider object pixels and thus positive examples.

We use these regions to compute two color histograms, which represent $P(f_c|object)$ and $P(f_c|\neg object)$ as suggested in [23]. Since we consider pixels as positive or negative examples now (as opposed to whole images as for the initial model explained in the previous section), we can simply use relative frequencies to define these conditional probabilities:

$$P(f_c|object) = \eta_P^{-1} |\{\mathbf{p} \in \mathfrak{P} | f_c(\mathbf{p}) = c\}| \tag{6.6}$$

$$P(f_c|\neg object) = \eta_N^{-1} |\{\mathbf{p} \in \mathfrak{P}_N | f_c(\mathbf{p}) = c\}| \tag{6.7}$$

where $\eta_P = |\mathfrak{P}|$ and $\eta_N = |\mathfrak{P}_N|$ are normalizing factors and $\mathfrak{P}_N$ are all pixels in all images of N. Analogous to [23], we decide whether a color f_c belongs to the object by the following inequality:

$$\frac{P(f_c|object)}{P(f_c|\neg object)} > \theta_c \tag{6.8}$$

with

$$\theta_c = T_c \cdot \frac{P(\neg object)}{P(object)} = T_c \cdot \frac{1 - P(object)}{P(object)} \tag{6.9}$$

The prior $P(object)$ is the probability that any given pixel is part of the object and thus also depends on the size of the object in the image. Again, this value has to be determined empirically. In the evaluation section below, we set $T_c = 1$ for creating Receiver Operating Characteristic (ROC) curves over θ_c by sweeping the value of $P(object)$. For all remaining experiments, where a fixed value is required, we empirically set $P(object) = 0.2$ (and thus $\theta_c = 4$).

Evaluating equation 6.8 for each color f_c in our color space F_c yields the final object color model h_c:

$$c_c(f_c) = \begin{cases} 1 & \text{if } \dfrac{P(f_c|object)}{P(f_c|\neg object)} > \theta_c \\ 0 & \text{otherwise} \end{cases} \tag{6.10}$$

According to equation 4.8, we define the set of positive colors $F_{c,pos}$ based on this model by

$$F_{c,pos} = \{f_c | c_c(f_c) = 1\} \tag{6.11}$$

Figure 4.2 shows an example for $F_{c,pos}$ (in RGB color space for better visualization). Based on the positive colors, we define our updated sets of positive pixels $B_c(i)$ for each image i:

$$B_c(i) = \{\mathbf{p} \in P_i | f_c(\mathbf{p}) \in F_{c,pos}\} \tag{6.12}$$

Note that we now use actual pixel frequencies instead of image frequencies which are used for the initial model. In figure 6.1c, the set $B_c(i)$ is visualized for an image (figure 6.1a) from the Oxford Flowers dataset [29]. Also, the positive pixels for the initial model c'_c are shown in 6.1b, i.e., before applying the flood fill algorithm and the relative frequency-based model explained in this section. In this example, we can clearly see that, even though c'_c already removes the majority of background pixels, the final color model c_c still improves the result by removing outliers.

Note that if we do not find many unique colors, the flood fill algorithm will also be seeded at background colors, since non-object colors will be among the top colors found by the initial model c'_c due to the dynamic threshold.

This causes the algorithm to also cover large background areas and thus "over detect" the wanted objects which is an important property we will use for the combined model explained in section 8.

(a) (b) (c)

Fig. 6.1: Example for color-based positive pixels $B_c(i)$ of image i (a) with initial model (b) and after flood fill-based refinement of our color model (c).

Based on $B_c(i)$ we can create bounding boxes as explained in section 5.1.2. Before creating bounding boxes, however, there are still some parameters which we need to evaluate such as the color space and the number of bins used for dividing this color space. Thus, in the following section we explain how we evaluate our color model and we examine these parameters. Also, we compare our model to the optimal results based on manual pixel-wise annotations.

6.3 Evaluation of Color Model

In this section we first explain how we determine feature-specific parameters of the color model such as color space and number of bins on the FlickrLogos-6 dataset. Afterwards, we evaluate the color model as a means to identify positive pixels. Finally, in section 6.4 we use the color model in our automatic object annotation scenario. Note that this section follows the evaluation of [31] where the threshold selection for the discriminative model is different.

6.3.1 Parameter Evaluation on Brand Logos

For our first set of experiments we use the FlickrLogos-6 dataset.

We use the recommended partitioning of 10, 30, and 30 images for the training, validation, and test set, respectively. FlickrLogos-32 also provides

a set of logo-free negative images which we use for training, too. For testing, we use $1,000$ random patches from World 100k as examples for negative pixels. For the following experiments we use the training and validation set combined to create the color model as described in the previous section and compute our results on the test sets.

For evaluating the quality of our color model, we use manual annotations of our datasets. In each positive image the pixels which belong to the logo were labeled manually. Thus, for each pixel p we have a ground truth label $t(\mathbf{p}) \in \{1, 0\}$. Despite careful labeling, the annotations contain some noise, i.e., annotations of the logos also may include some background pixels.

Method of Evaluation

For all of the following experiments on the FlickrLogos-6 dataset we use the same method of evaluation. We measure the effectiveness of our approach by constructing a ROC curve over the respective threshold θ_c from equation 6.10. The ROC curve requires a *true positive* (TP) rate as well as a *false positive* (FP) rate for each tested threshold value. Thus we simply determine among all positive test images the fraction of pixels with $t(\mathbf{p}) = 1$ whose color $f_c(\mathbf{p})$ our respective color model classifies as an object color (i.e., $c_c(f_c(\mathbf{p})) = 1$) in order to obtain the TP rate. Note that for each pixel $\mathbf{p} \in \mathfrak{P}_N$ of any negative image we know that $t(\mathbf{p}) = 0$ since the negative images never show a logo. Therefore, we simply sample $1,000$ random rectangles with size 200×200 pixels from the negative images as negative examples. We then determine the FP rate as the fraction of pixels among these rectangles whose color $f_c(\mathbf{p})$ is classified as an object color.

Note that for clarity, we do not interpolate from the final pair of true positive rate and false positive rate to the pair $(1.0, 1.0)$ which is obviously the final point of each ROC curve, since we can always simply apply a decision rule which accepts every pixel as positive. We let our curves stop at their maximum TP rate and before the TP-FP pair $(1.0, 1.0)$.

Our experiments and results differ from [31] for two reasons. First, our initial model, which is designed for generic features, is not optimized specifically for the color feature as opposed to the method of [31]. As a consequence, we do not get comparable results. Second, we use a significantly smaller negative set.

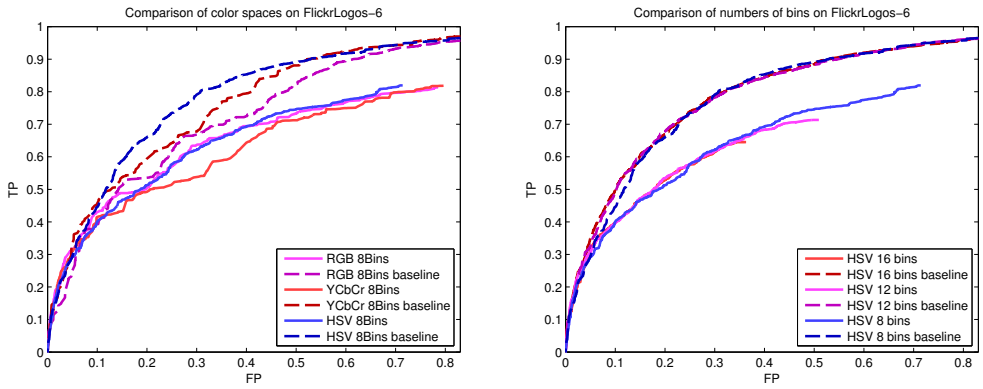
We do not count the FP among positive images since the logos are often found on surfaces of the same color as the logo (e.g. billboards, wrappings,

or vehicles) which were not annotated as belonging to the logo. However, the pixels of these surfaces cannot be considered actual false positives. Thus, the annotation in positive images is ambiguous regarding negative pixels since the logos were simply annotated as narrowly as possible.

Comparison of Color Spaces and Numbers of Bins

We compare the performances of our model in three different color spaces: RGB, HSV, and YCbCr. For this experiment, we divide each color channel into 8 equally sized bins.

We also implement the approach of Jones and Rehg [23] which follows equations 6.6 to 6.10 whereas for the set of positive pixels $\mathfrak{P}$ our manually annotated positive pixels are used. Note that this approach is based on pixel-wise ground truth (except for some inaccuracies of the manual annotations). In fact, for the histogram approach, we obtain the optimal possible result if we use the manual annotations. Our goal is therefore to deduce a model which is as close to this method as possible without having to rely on manual annotations.



(a) Color spaces *(b)* # bins per channel

Fig. 6.2: Average ROC curves over all six logo classes over θ_c for (a) three different color spaces; (b) three different numbers of bins per channel in HSV color space. The results of using manual annotations is labeled as "baseline".

We compute a combined ROC curve over all 6 logo classes. The combined ROC curves for RGB, HSV, and YcbCr color spaces are shown in figure 6.2a as well as the corresponding ROC curves for the average results of the optimal

approach. Note that all three color spaces do not go beyond a true positive rate of about 0.8 which is, however, usually not necessary for locating objects.

On average, the HSV color space outperforms YCbCr color space and yields a slightly better performance than RGB. Up to a false positive rate of about 0.75, RGB is on par with HSV, however, the optimal approach is considerably better for HSV than for RGB, which supports picking HSV space. We hence conduct all remaining experiments in HSV color space.

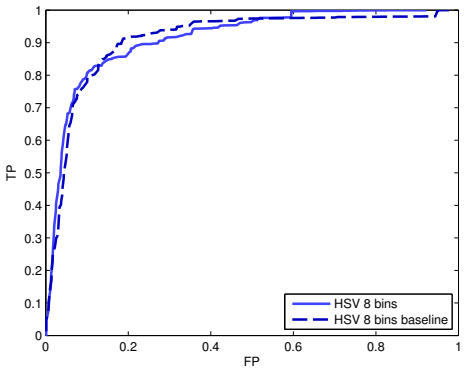
After determining the best color space, we evaluate different numbers of bins per channel. Again, we determine the TPs and FPs over all positive and negative test images for 6 logo classes and average the results. The resulting ROC curves are shown in figure 6.2b. We compare results for 8, 12, and 16 bins.

According to the ROC curves, the results are almost identical for all numbers of bins up to the points where we do not find any more true positive pixels. With 8 bins we find the largest ratio of true positives. Another advantage of 8 bins is that we get a more compact histogram, so we use 8 bins per channel in HSV color space for the remainder of this work.

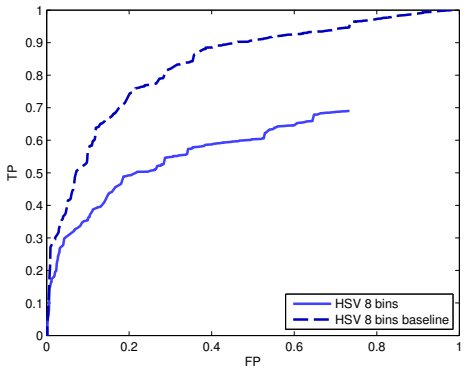
Comparison to Optimal Method For Logos

After determining the histogram configuration, we examine the performance of our approach on each of our 6 logo classes for this configuration. As a consequence of the previous experiments we use the HSV color space and 8 bins per color channel. We conduct the same experiment as described above except for averaging over all classes. Instead we determine the ROC curve for each class separately. The results are shown in figure 6.3.

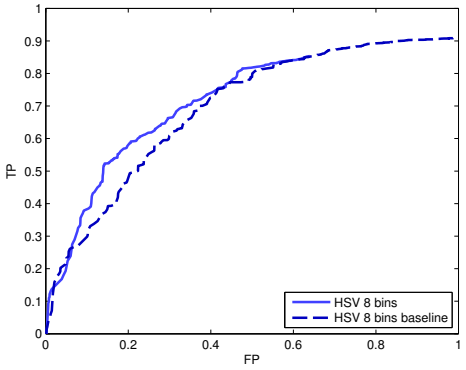
Again we include the results of the optimal approach in each graph. Overall, our approach is relatively close to the optimal method for all classes except for "Aldi" and "Pepsi". For both classes, we obviously miss a relatively large number of positive pixels, since the curves stop around 0.6 and 0.7 true positive rate, respectively. The reason for the exceptionally large difference on "Aldi" is that the "Aldi" logo is (with regards to color) relatively difficult since it consists of four different main colors. Besides, the "Aldi" logo is very small in many training images which also applies to the Pepsi logo. As the qualitative examples in figures 6.4 and 6.5 suggest, we miss relatively large parts of both object classes.



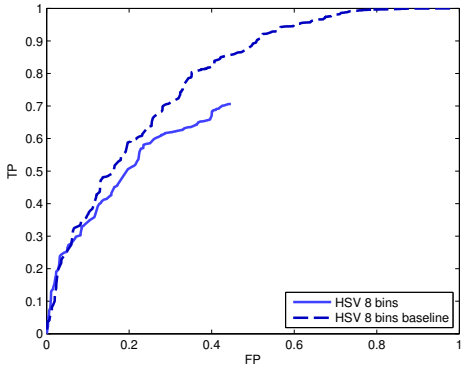
(a) Class "DHL"



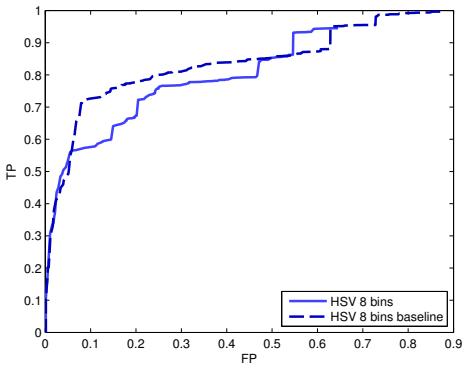
(b) Class "Aldi"



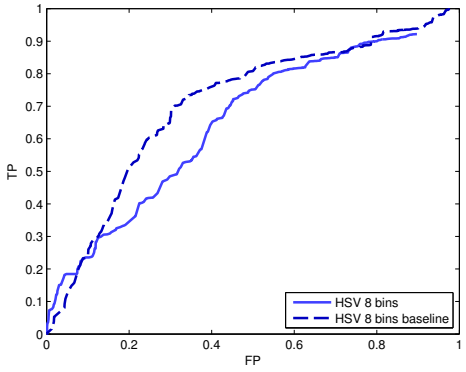
(c) Class "Coca Cola"



(d) Class "Pepsi"



(e) Class "Shell"



(f) Class "Esso"

Fig. 6.3: ROC curves over θ_c for our approach (solid line) and optimal approach [23] (dashed line) in HSV space for 8 bins per channel.

However, we still obtain indicative colors for these classes which lead to bounding boxes which overlap with the wanted object instance. Since our initial model is supposed to be applicable to a large number of classes and two different features, we refrain from tweaking the model only to achieve a higher true positive pixel rate for these classes.

Interestingly, both the optimal and our approach work best for the class "DHL" which is due to the fact that the "DHL" logo usually contains bright yellow which is quite uncommon in background images. This is also true for the class "Shell" for which we also obtain good results. Another interesting observation is that our approach yields slightly better results than the optimal method on the "Coca Cola" class. The "Coca Cola" logo consists of two colors, namely red and white. Intuitively speaking, our statistical model identifies red as an exceptionally strong "best feature" f_c^* which causes white (a color relatively often present in background) to fall into the confidence interval for background colors. Dismissing white is, however, beneficial for the ROC curve since it does not constitute a major part of the logo (i.e., the positive pixels) but appears relatively often on negative images.

Figures 6.4 and 6.5 show one example test image for each logo class and the resulting image after removing all pixels which are classified as negative by the color model. For these examples we used our default values $p(object) = 0.2$ and $T_c = 1$ in equation 6.10, i.e., $\theta_c = 4$. Most of the non-logo pixels are correctly removed in the resulting images, however, pixels which have the same color as the logo obviously cannot be excluded by a color model. Since we set our parameters conservatively, we also miss some logo pixels.

6.3.2 Evaluation on Flowers Dataset

The second dataset we use for evaluation is a subset of the Oxford Flowers dataset [29] which consists of 17 flower classes with 80 images each. For consistency with the previous experiments, we again use the HSV color space with 8 bins per channel for our experiments. The flower dataset poses a slightly different task than the brand logos, since flowers are natural objects and thus are subject to more variance with regards to appearance.

Some of the images are annotated pixel-wise, so we can use them to train the optimal approach and evaluate our method as described before. For comparability, we also only use annotated training and validation images for creating our color model. Some classes come with only very few annotated images (in one case none), so the number of training images varies from 0

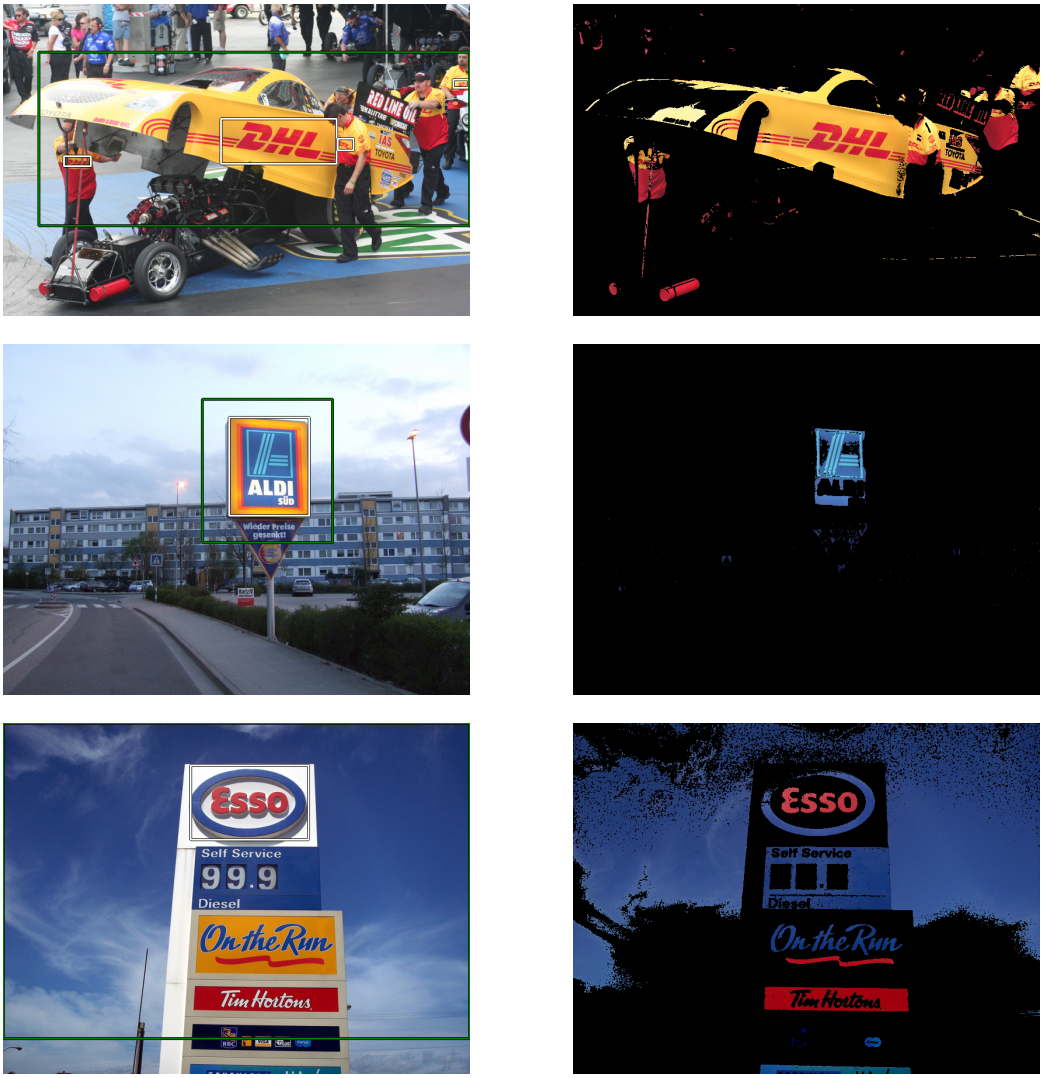


Fig. 6.4: Example results for 3 logo classes from FlickrLogos-6. Identified negative pixels were replaced by black pixels in the right image of each pair. The remaining pixels are the respective color-based positive pixels $B_c(i)$. In the left image of each pair, the resulting bounding box is shown. Also, the ground truth annotation is shown by a white rectangle.

to 53. We omit the two classes which only have 10 and 0 annotated training images. Note that we keep one class which has only 20 training images, even

Fig. 6.5: Example results for 3 logo classes from FlickrLogos-6. Visualization is analogous to figure 6.4.

though this is in theory considered not sufficient for estimating a confidence interval. Since the flower dataset does not contain a negative set, we simply use all images from all classes except the positive class as negative training images.

We conduct the same experiments as for the logos to obtain TPs and FPs values for each class. However, since the flowers are annotated unam-

biguously, we can compute the FPs on the background of the positive test images. Again, the number of test images varies between 8 and 20 due to the limited number of available annotations.

We then also create a ROC curve over all classes which is shown in figure 6.6 on the left. Here, the optimal approach works significantly better than our approach. This observation is caused by four classes for which our model completely fails, i.e., actually models the background.

Three of these classes are white flowers. White, however, is a color which is not determined as an indicative color by our discriminative model, since it is often present in the background. As is typical for most flower classes, the background has relatively little variance in the positive images and in the face of positive colors with little discriminativeness (i.e., white), background colors are then determined as more discriminative than the foreground. Another class for which our model fails is "Iris" which also has many white instances and the remaining instances do not have a fixed color scheme. The instances appear in about five different main colors and various combinations while the background is considerably less variable. Both cases violate one of our assumptions, i.e., the objects either do not have a feature which distinguishes it from background or do not have a consistent feature representation across the majority of positive images.

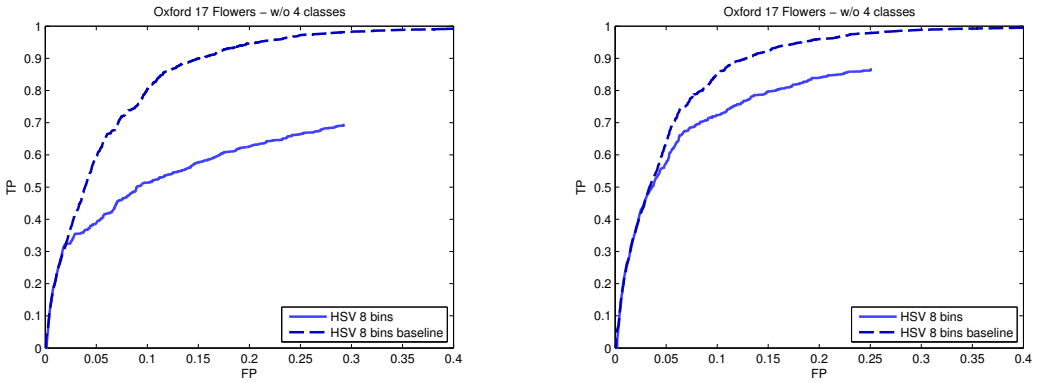


Fig. 6.6: ROC curves over positive and negative pixels on Oxford 17 Flowers dataset. Left: All classes, Right: without four classes violating our assumptions.

In order to illustrate these observations, we also show the results without these four classes on the right of figure 6.6. The resulting ROC curve for our

approach is then much closer to the optimal approach.

Note that some of the remaining classes still feature flowers, which occur in varying colors in different images, e.g. "Pansy" or "Fritillary". Thus we consider this dataset more challenging than the logos dataset due to the higher amount of intra-class variance. Also note that many flower classes are very similar and almost identical regarding the color scheme. Therefore object colors in some cases appear in a considerable number of background images. Yet a little amount of noise among the negative images apparently does not affect the performance of our approach in the majority of cases, which further eases providing negative sets in practice.

A few qualitative example results are shown in Figure 6.7. Interestingly, tones of green are also determined as positive colors which is due to the fact that the backgrounds of the flower images lack variance. Also note that for the first example, we miss almost half of the positive pixels which, however, does not negatively affect the resulting bounding box.

6.3.3 Application to Skin Detection

One important application of discriminative color models is the detection of human skin. Note that an important application of a skin color model is the filtering of adult image for which is in particularly useful if a model can be devised without manual annotations. An overview of different methods for adult image filtering can be found in [32]. Many are based on color models.

For this experiment we use the Bikini dataset which consists of images showing people in swim wear. Thus the amount of skin colored pixels is relatively high among these images. However, since pictures of people in swim wear are usually taken outside and close to water, the Bikini dataset features a limited number of fundamentally different backgrounds. Still the main common property of all images is the presence of human skin.

Our negative set is the random background set World 100k. Recall that World 100k also contains images showing people, however, these people are usually not wearing swim wear, thus visible skin areas tend to be small.

For creating our model, we use $1,000$ images from the Bikini dataset and $6,000$ images from the World 100k dataset as positive and negative images, respectively. Figure 6.8 shows the results of applying our color model to some images of the Bikini dataset where the detected positive pixels (which correspond to human skin) are highlighted. A few negative images from World 100k where typical false positive pixels were found are shown in figure 6.9.

Fig. 6.7: A few example results from several classes of the Oxford Flowers dataset. Visualization is analogous to figure 6.4.

Since we do not have pixel-wise annotations for the Bikini dataset, we cannot provide quantitative evaluation as for the previous datasets. However, in the next section, the color model will be evaluated quantitatively for detecting bounding boxes in the Bikini images.

Fig. 6.8: Example images for true positive pixel detections on positive images and resulting bounding boxes for skin color model on Bikini dataset. Visualization is analogous to figure 6.4.

6.4 Evaluation of Color Model for Automatic Object Annotation

After showing the accuracy of our color models on pixel-level, we now evaluate to which degree our estimated bounding boxes match the actual respective object locations. We therefore evaluate our color model with regards to overlap with ground truth bounding boxes by plotting overlap-recall curves as explained in section 5.2.

Note that for now we concentrate on finding only one bounding box for each positive image. We will examine the multi-object scenario in chapter 9. Further note that annotating only one object instance per positive image is also a sound application if the main purpose of our approach is not detect-

Fig. 6.9: False positive pixel detections on negative images for skin color model. Again, in the right image, all pixels are replaced by black pixels except positive pixels $B_c(i)$.

ing all objects but to create initial training examples for machine learning algorithms. Our goal is thus to estimate a single rectangular bounding box $\hat{r}_i$ for each image i in the positive image set.

It is important to mention that we adapt the color space and number of bins from the previous experiment on FlickrLogos-32 which uses manual annotations for evaluation. One could hence argue that we violate our assumption for automatic annotation that we do not have any additional knowledge besides the global labels. However, one can view the previous experiment as a separate problem with a different evaluation protocol, which could also be performed on an unrelated dataset. Also, according to the aforementioned results, the different color spaces yield relatively similar results and we do not further optimize the parameters for each single dataset. Overall, it is thus justifiable that choosing HSV and 8 bins (which has the advantage of resulting in a very compact histogram) can be considered independent choices.

6.4.1 Brand Logos

Even though the ROC curves for the logos in figure 6.3 suggest that our color models are fairly accurate, we cannot deduce that we will find good bounding boxes based on color alone. The reason is, as stated above, that especially for the logo classes, the positive colors usually also appear in large background areas which leads to over detection and thus to low overlap values.

This issue can best be seen in the first example of figure 6.4 where despite a reasonable color model, the positive pixels include a much larger area than the actual logo. In fact, this problem is especially pronounced for the "DHL" class where almost every estimated bounding box includes the actual logo instance and at the same time removes a considerable amount of background but still obtains a poor overlap value due to similarly colored background. Other examples show unrelated (i.e., "actual") false positive pixels in the background which also have positive colors, i.e., the third example in figure 6.4 and the first example in figure 6.5. As a consequence, the overlap-recall curves shown in figure 6.10 are relatively poor for most classes despite reasonable color models. In other words, color is not discriminative enough for this dataset. Still, a certain amount of object instances is detected with a high overlap for almost every class and the majority of instances is at least included in a bounding box with an overlap > 0.

6.4.2 Flowers

Naturally, flowers usually do not appear in front of backgrounds of the same color as the flower itself. Thus, the aforementioned issue which leads to poor overlap-recall curves on the logo classes is not as severe for the flower classes. This is reflected by the overlap-recall curve in figure 6.11. We plot all 17 flower classes in one single overlap-recall-curve, i.e., all images of all classes are shown and their respective overlaps sorted descending from left to right. For consistency with the upcoming experiments, we now also include the classes for which the color model cannot be expected to work as explained above. Still one flower class is skipped since it does not provide any annotated examples. The results are considerably better than for the logo classes. A few qualitative examples can be seen in figure 6.7.

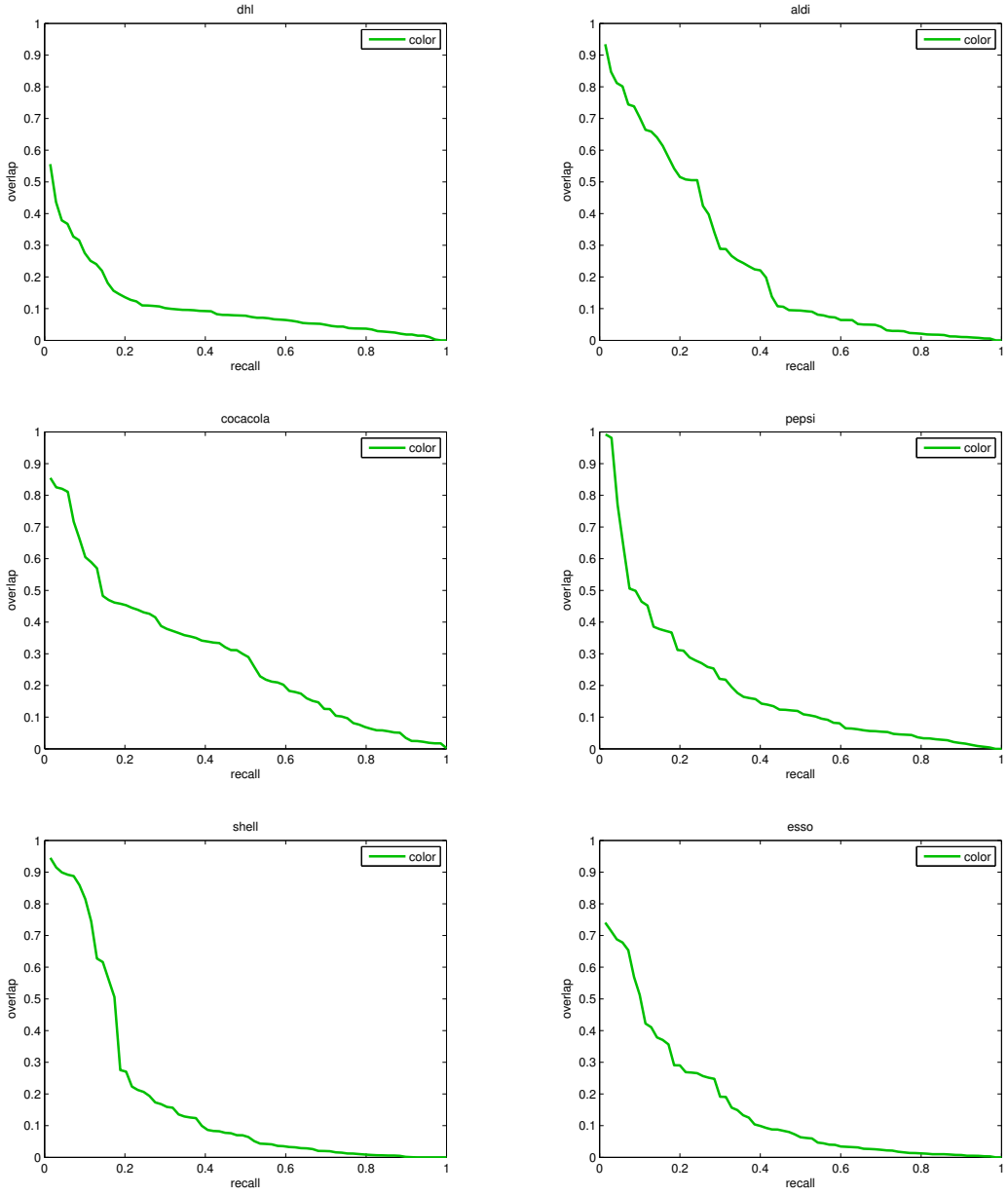


Fig. 6.10: Overlap-Recall curves for each class from FlickrLogos-6, based on color.

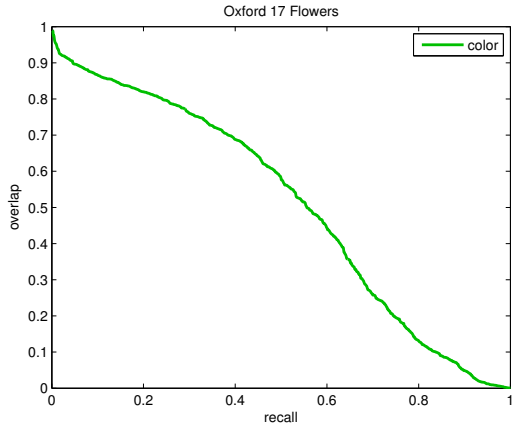


Fig. 6.11: Overlap-recall curve for the inferred color model on all flower classes combined.

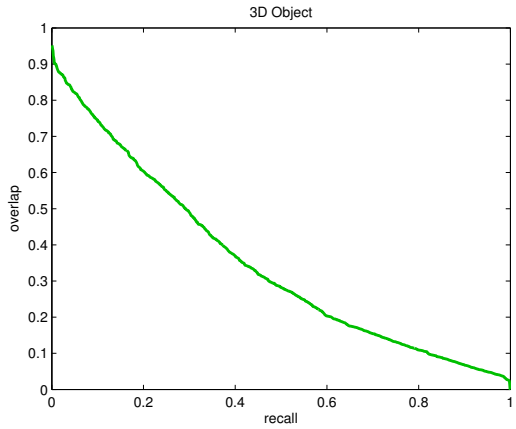


Fig. 6.12: Overlap-recall curve for the inferred color model on 3D Object Categories dataset.

6.4.3 3D Object Categories

Finally, we evaluate the color model on the 3D Object Categories dataset. The results are shown in figure 6.12 in one plot over all ten classes. Even though the curve is lower than the curve for the flowers dataset where color is exceptionally well suited, the color model is still capable of removing background areas for a number of images. The reason is that the intra-class

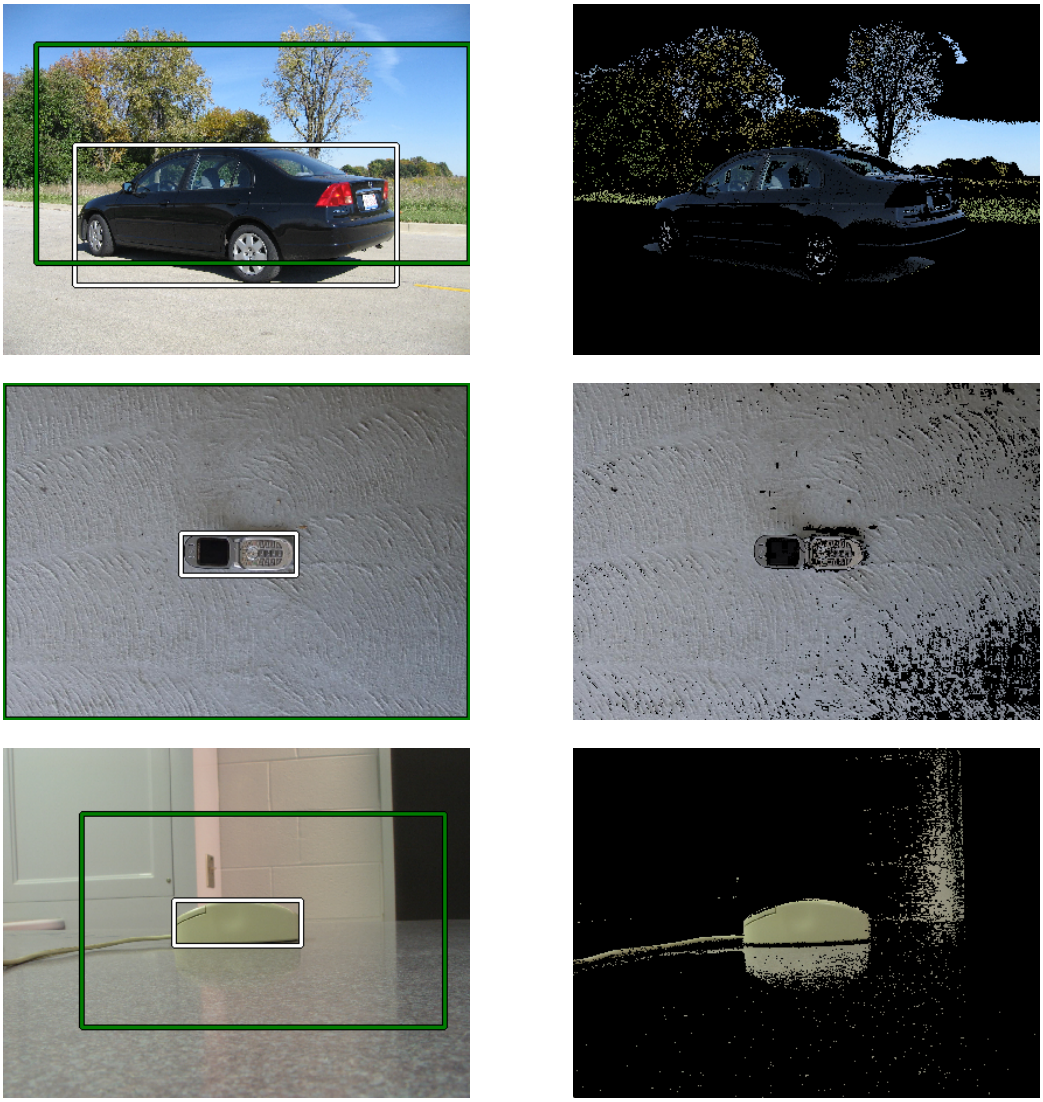


Fig. 6.13: Example results for the color model on three classes from 3D Object Categories dataset.

variance with regards to color is not as high as one might expect, since, for instance, the car class does not contain a large number of differently colored instances. Besides, other classes have a few indicative colors like "head" because of consistently colored skin and hair or "bicycle" because of consistently

colored spokes.

Figure 6.13 shows a few example images where the original images are shown along with binary masks of positive pixels found by the color model. Also, the respective resulting bounding boxes are shown in the original images. For the first example, the color model includes background colors which are regularly present in images from the "car" class. The resulting bounding box, however, still excludes some background and true positive colors from the car are also included in the color model, i.e., we apparently have a relatively large set of positive colors. In the second example, the color model considerably over detects the desired object, since the object has roughly the same color as the background. In the final example, the color model actually determines the pixels of the desired object relatively accurately since for the "mouse" class, the colors of many instances are somewhat consistent.

6.4.4 Bikini

In this section, we show the overlap-recall curve based on our skin color model obtained from the Bikini dataset. The curve shown in figure 6.14 is created on the annotated portion of the training set which includes $1,000$ images.

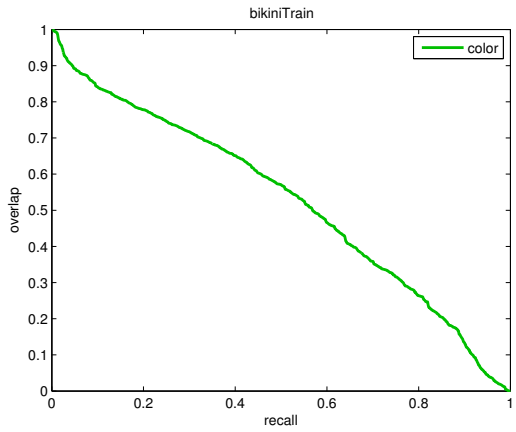


Fig. 6.14: Overlap-recall curve for color model on Bikini dataset.

Since skin color is a strong indicator, the color-based overlap-recall curve shows reasonable performance given the difficulty of the dataset and the strictness of the overlap measure. It, however, suffers from relatively low variance in the background and also groups of people within single images

which lead to overly large bounding boxes. Also, skin color is more common in background areas than many flower or logo colors (see figure 6.9 for a few examples).

7. HOG MODEL

In the previous chapter we have explained how we instantiate our discriminative model for color features. Colors are naturally a straightforward visual feature. However, they are often not very discriminative. The main problem is that often background objects also include colors of the wanted objects. For instance, in the first example in figure 6.4, the car has the same color as the wanted logo and thus cannot be discriminated from the logo based on color alone.

Therefore we introduce a complementary local feature, which is based on image gradients which are independent from actual color values (in a chromatic sense) and model structures and textures. There are several local image features which encode gradient information of a given image patch into a fixed-length feature vector, for instance SIFT [25] and SURF [4]. Among the most popular gradient-based features are Histograms of Oriented Gradients (HOG) as introduced by Dalal and Triggs [12] which we use for our discriminative model. HOG features have been successfully used for object detection [20] and are in particular suitable for dense computation on a regular grid. Also, they can be computed fairly rapidly. Our HOG implementation follows [20]. We also utilize the Bag-of-Visual-Words paradigm [42] in order to obtain a finite set F_{hog} of HOG features which is a prerequisite of our approach.

7.1 Multi-Scale HOG

The main idea of the HOG feature is to encode the appearance of cells of a fixed-sized grid by creating a histogram over the gradients found within these cells. Gradients are defined by their magnitude and their orientation. The histogram is built over the orientations which are binned into nine disjoint bins. For each gradient within a cell, its magnitude is proportionally added to the respective orientation bin. In the basic formulation, only the interval $[0, \pi)$ is considered for binning. Gradients with orientation $\alpha + \pi$ account

(a) (b)

Fig. 7.1: Visualized HOG features in (b) of a given image in (a).

for the same bin as gradients with orientation α, since such gradients only differ with regards to their sign and thus describe edges with complementary angles.

In figure 7.1, a HOG example is visualized. For each of the nine bins of each HOG cell, we draw an edge which is orthogonal to the respective gradient direction (since gradient directions always point "across" edges). The brightness of the edge represents the magnitude of the respective bin. For better visualization, the magnitudes are normalized over all cells. The shape of the airplane can clearly be recognized in the "HOG image". It can also clearly be seen that HOG features in "empty" image regions are often highly noisy (even though in this image the effect is amplified by normalization over all cells).

A common variant of the HOG features, which we also use, is described by Felzenszwalb et al [20]. The first 9 bins of this variant of the feature also correspond to the main gradient orientations. Then another 18 bins are added which correspond to the gradients binned according to the full angular interval $[-\pi, \pi)$, i.e., the sign of the gradient is considered. Hence we obtain 27 values per cell. Finally 4 values are added which represent the overall gradient magnitudes of all four possible blocks of 2×2 cells which include the respective cell for which the feature is computed. This variant of the HOG descriptors is explained in detail in [20].

Since the HOG descriptor counts the relative number of gradient orientations found within an image region, it is to some extent invariant against scaling. It is not invariant against rotation except for the little invariance

introduced by the discretization of gradient orientations. Full rotational invariance, however, is not desirable for many objects which usually appear in a limited number of rotational variants in images.

As single HOG cells have relatively low descriptive power, we concatenate four neighboring HOG cells arranged in a 2×2 assembly into one feature descriptor. Thereby we obtain a set of overlapping features which weakly encode the spatial layout of multiple cells. This allows describing more complex primitive shapes than mere straight lines. We also adopt the standard cell size of 8×8 pixels.

Since the standard cell size only covers objects at one certain scale, we extract the HOG features on multiple scales. We therefore build an image pyramid as for instance explained in [25] by scaling the image repeatedly with a factor of $2^{-0.25}$. This process is visualized in figure 7.2. Since the HOG cell size remains 8×8 pixels, potential objects are cpatured at different absolute sizes. When projected back into the original images, the cells at small image scales describe larger areas.

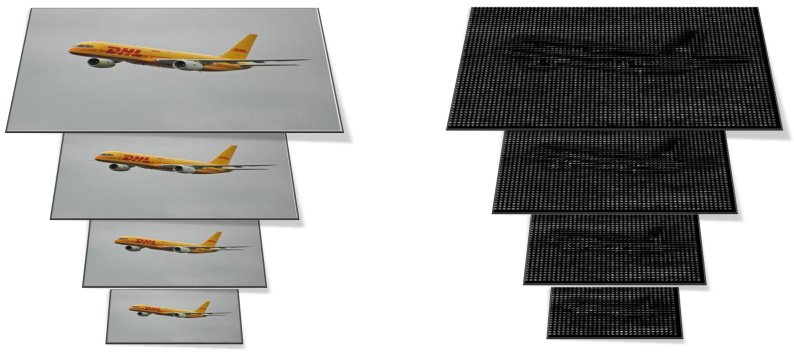


Fig. 7.2: Visualization of image pyramid for multi-scale HOG extraction. The original image (left) is scaled down repeatedly and for each scale HOG features are extracted separately.

Note that for our statistical model, we need a finite set of features. Therefore we use a Bag-of-Words model as described in the following section.

7.2 Feature Quantization

For quantizing our HOG features into a finite set, we apply the Bag-of-(Visual)-Words (BOW) paradigm. The BOW approach [42] to object detection is based on the idea that objects are visually composed of a limited number of visual primitives. Analogous to texts which are composed of a limited number of different words, images are assumed to be composed of so-called *visual words* from an finite *visual vocabulary*. A vocabulary of visual words is built by extracting a large number of local features from a given image set and quantizing them into a fixed number of clusters, e.g. by the standard *k-means* clustering algorithm. Thus, visually similar local features will fall into the same cluster and are interpreted as the same visual word. This reduces the dimensionality of one local visual feature to 1, since we only need the cluster number (the so-called "cluster id") to identify a visual word.

Interestingly, the image set which is used for creating the vocabulary may be a random set of images which does not have to prominently feature the wanted object [34] under certain circumstances. Thus, for creating the visual vocabulary we do not necessarily need to know the positive images, which is useful if the positive images are not easily accessible. However, this is only true if the random set of images is diverse enough to produce "universal" visual words and if the target objects are not overly specific, i.e., if they do not require visual words which are not present in a random set of images. Thus, for highly specific classes like brand logos which feature structures which are not commonly present in other image sets, it can still be considered beneficial to extract the vocabulary from the given dataset which we do for our experiments.

7.3 Discriminative Model

Now that we can describe each 2×2 HOG feature by its one-dimensional cluster id f_{hog}, we can define a finite set F_{hog} of HOG features:

$$F_{hog} = \{f_{hog} | f_{hog} \in \{0, 1, ..., k-1\}\} \tag{7.1}$$

where k is the number of visual words. We experimentally determine that $k = 10,000$ is a good compromise between expressiveness of the individual words and compactness of the feature space. Analogous to before, we find the set of positive HOG features $F_{hog,pos}$ by applying our discriminative model

to each feature based on occurrence frequencies following the steps explained in chapter 4 leading to equation 4.8:

$$F_{hog,pos} = \{f_{hog} | c_{hog}(f_{hog}) = 1\} \tag{7.2}$$

where c_{hog} is the function from equation 4.5 implemented for clustered HOG features. Note that, since the color model may already exclude background areas, we do not consider HOG features which are outside of the bounding boxes returned by the color model. Thus, if the respective object class has strong color features, we may already obtain less negative HOG features from positive images. Otherwise, the color model will tend to return over-detections which are very likely to still include the positive HOG features. In figure 4.2 a few examples from a set $F_{hog,pos}$ are shown, each by four selected HOG patches falling into a positive BOW cluster.

Now we need to define the association between a visual word f_{hog} and corresponding pixel locations, i.e., we need to determine all features $F_{hog}|_{\mathbf{p}}$ associated with a given pixel $\mathbf{p}$. Our HOG grid does not have points at every possible pixel location, therefore this association is not as straightforward as for colors. Each HOG feature corresponds to a patch which covers multiple pixels, thus each pixel is covered by multiple HOG cells on our scale pyramid. We hence have a many-to-many relation between pixels and features.

First, let $h_{\mathbf{p}'}$ be the HOG descriptor extracted at pixel $\mathbf{p}'$ and $s_{h_{\mathbf{p}'}}$ the scale on which it was extracted among our scale pyramid. The patch radius $w(h_{\mathbf{p}'})$ of HOG descriptor $h_{\mathbf{p}'}$ is defined by

$$w(h_{\mathbf{p}'}) = s_{h_{\mathbf{p}'}} \cdot w_{hog} \tag{7.3}$$

where w_{hog} is the basic HOG cell size (8 pixels). Note that we use 2×2 HOG cells as features, thus the size of one cell is the "radius" of one HOG descriptor. Now, let $f_{hog}(h_{\mathbf{p}'})$ be the visual word corresponding to HOG descriptor $h_{\mathbf{p}'}$ whose patch is centered at $\mathbf{p}'$. Then, we can define

$$F_{hog}|_{\mathbf{p}} = \{f_{hog}(h_{\mathbf{p}'}) | d(\mathbf{p}', \mathbf{p}) < w(h_{\mathbf{p}'})\} \tag{7.4}$$

where $d(\mathbf{p}_1, \mathbf{p}_2)$ is defined as

$$d(\mathbf{p}_1, \mathbf{p}_2) = \max(|x_p - x_{p_h}|, |y_p - y_{p_h}|)\} \tag{7.5}$$

Thus, if pixel $\mathbf{p}$ is covered by a HOG descriptor's patch and the descriptor's corresponding visual word is f_{hog}, $\mathbf{p}$ is associated with f_{hog}. Given our def-

inition of $F_{hog}|_{\mathbf{p}}$, we can finally determine the visual words-based positive pixels $B_{hog}(i)$ of image i as already defined in equation 5.1:

$$B_{hog}(i) = \{\mathbf{p} \in P_i | F_{hog}|_{\mathbf{p}} \cap F_{hog,pos} \neq \emptyset\} \tag{7.6}$$

7.4 Evaluation

As for the color models, we again create overlap-recall curves for the HOG models. We use the same datasets and evaluation procedure as explained in detail in sections 5.2 and 6.4.

7.4.1 Brand Logos

The results on our classes from the FlickrLogos-6 dataset are shown in figure 7.3. For creating the discriminative model, we use the full images from the negative set of FlickrLogos-32 and add 5 random sub-images per image from random image scales. We therefore obtain 6 examples from each negative image resulting in a total of $36,000$ negative examples.

Interestingly, the visual words-based features yield similar overlap-recall curves for almost all logo classes compared to the color model. This is due to the fact that logos are planar, rigid objects which usually are not subject to much variations in real-world photos except different absolute scales which is mostly compensated by the multi-scale HOG approach. Still, HOG features are usually subject to noise, especially in the form of false positive features, as for instance thoroughly discussed in [49]. Thus, color and HOG features are relatively similarly suited for this dataset. The only exception is the "Aldi" logo, where the HOG model detects inferior bounding boxes since it tends to over detect due to structures in the background which either are similar to parts of the logo or appear in a relatively high number of positive images (the "Aldi" images often have similar backgrounds). Also, the "Aldi" logo is often relatively small in images which is not a problem for the pixel-level color model but for the HOG feature which has a minimum size of 16×16 pixels (8 pixels side length at 2×2 layout).

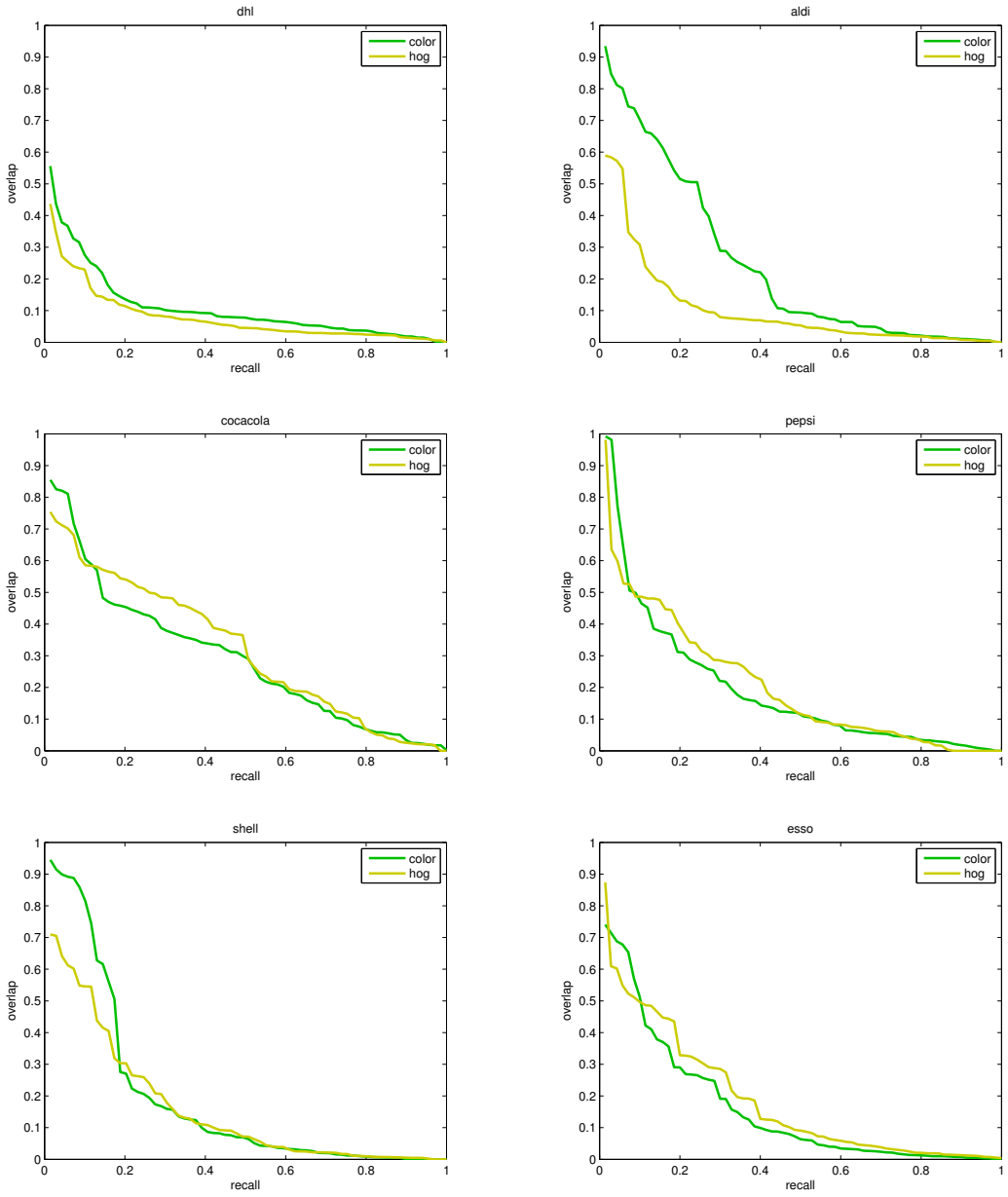


Fig. 7.3: Overlap-recall curves for each of our logo classes, based on visual words (labeled "hog") and color for comparison.

Fig. 7.4: Example results for three logo classes from FlickrLogos-6 ($B_{hog}(i)$ visualized on the right for each image). In the left image of each pair, the resulting bounding box is shown in yellow.

Again, we show a few qualitative examples in figures 7.4 and 7.5. These results reflect the overlap-recall results. While for instance the "Esso" logo can be found almost perfectly based on HOG features for the given example image, the "Aldi" and "DHL" logos are detected with large background areas

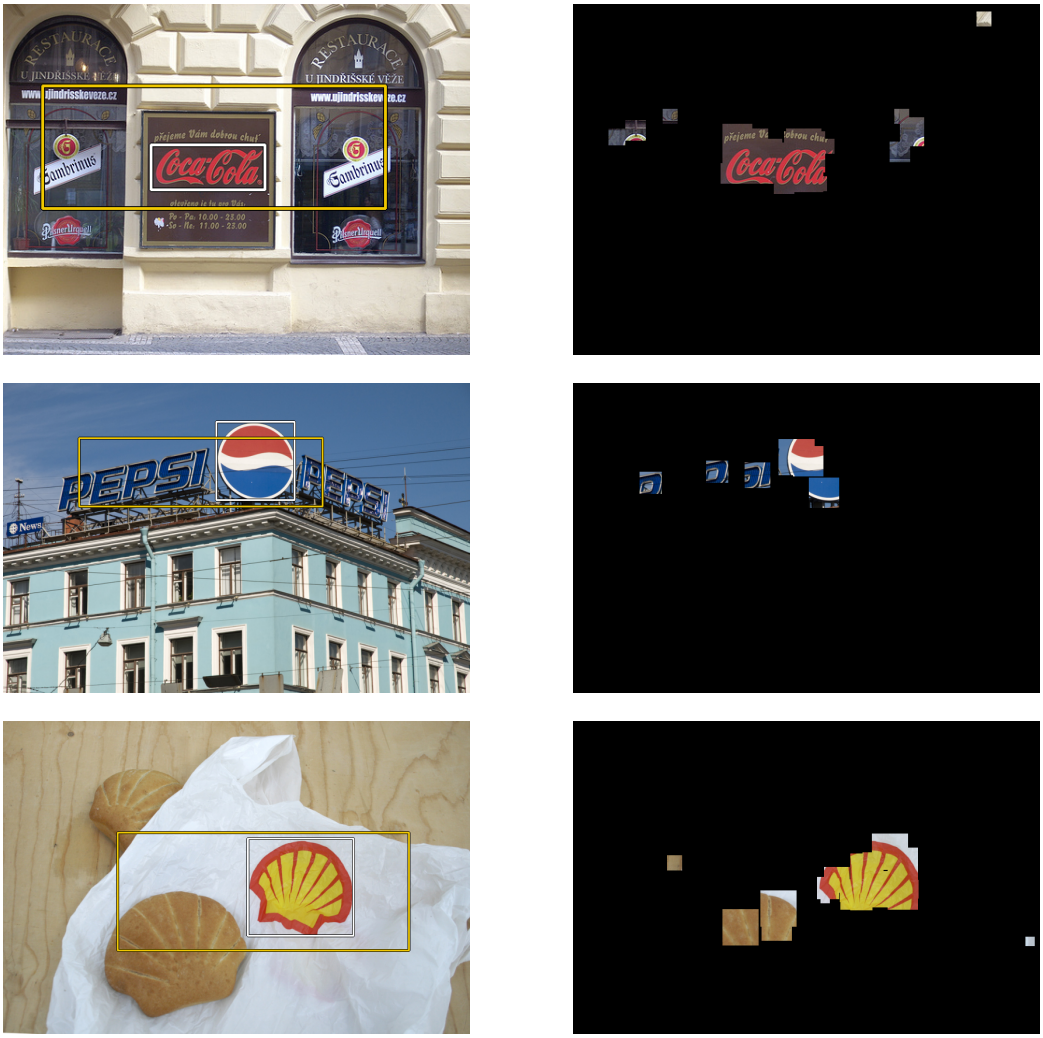


Fig. 7.5: Example results for three logo classes from FlickrLogos-6. Visualization is analogous to figure 7.4.

leading to bounding boxes with poor overlap values. However, note that these bounding boxes still contain the logo.

7.4.2 Flowers

For the Oxford 17 Flowers dataset, we also show the overlap-recall curve based on HOG features (again for all classes combined) in figure 7.6. The

HOG-based curve is worse than the color curve which indicates that HOG features are a less useful feature for this dataset compared to color features.

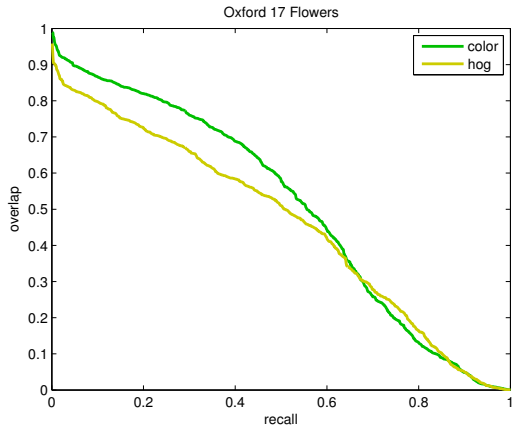


Fig. 7.6: Overlap-recall curve for Oxfrod 17 Flowers, based on visual words and color for comparison.

This observation is confirmed by the illustrative examples in figure 7.7. Gradient-based features are less unique than the respective flower's colors and therefore the discriminative model yields a relatively large set of positive features. For instance, in the first example the stem of the flower is also considered a positive area, since it is very similarly shaped as the flower itself with respect to gradients (and it is obviously present in many positive images). This leads to the inclusion of larger background areas compared to the color model.

Recall that with regards to the combined model explained in section 8, however, we designed the model such that in many cases it tends to over detect the wanted objects if our features are non-distinctive. Overly large detections do not affect the combined model negatively if the respective complementary feature (i.e., color) is superior.

7.4.3 3D Object Categories

In contrast to the Flowers dataset, the 3D Object Categories dataset intuitively has stronger gradient-based features then color features. Surprisingly, the gradient-based HOG model, however, does not outperform the color model as reflected by the overlap-recall curve in figure 7.8. The first

Fig. 7.7: Example results on Oxfrod 17 Flowers dataset. Visualization is analogous to figure 7.4.

reason is that, as discussed above, the color feature performs reasonable for many example images, since some classes have relatively consistently colored instances. The second reason is that some classes still have a relatively large intra-class variance with regards to the HOG features which leads to a large number of false positive features. Also, the image quality is relatively low

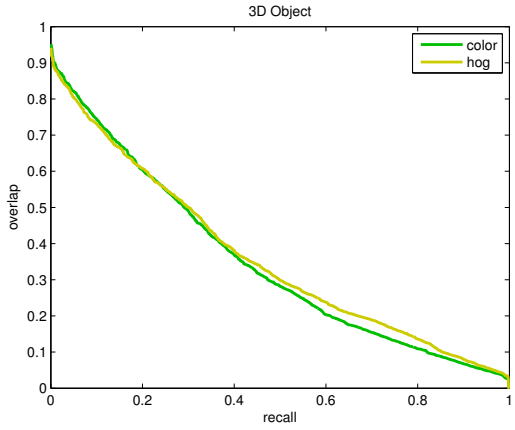


Fig. 7.8: Overlap-recall curve for color model and hog model on 3D Object Categories dataset.

resulting in a relatively large amount of noisy HOG features. The qualitative example results in figure 7.9 show that, while true positive regions are found, a considerable amount of false positive patches is present for all examples.

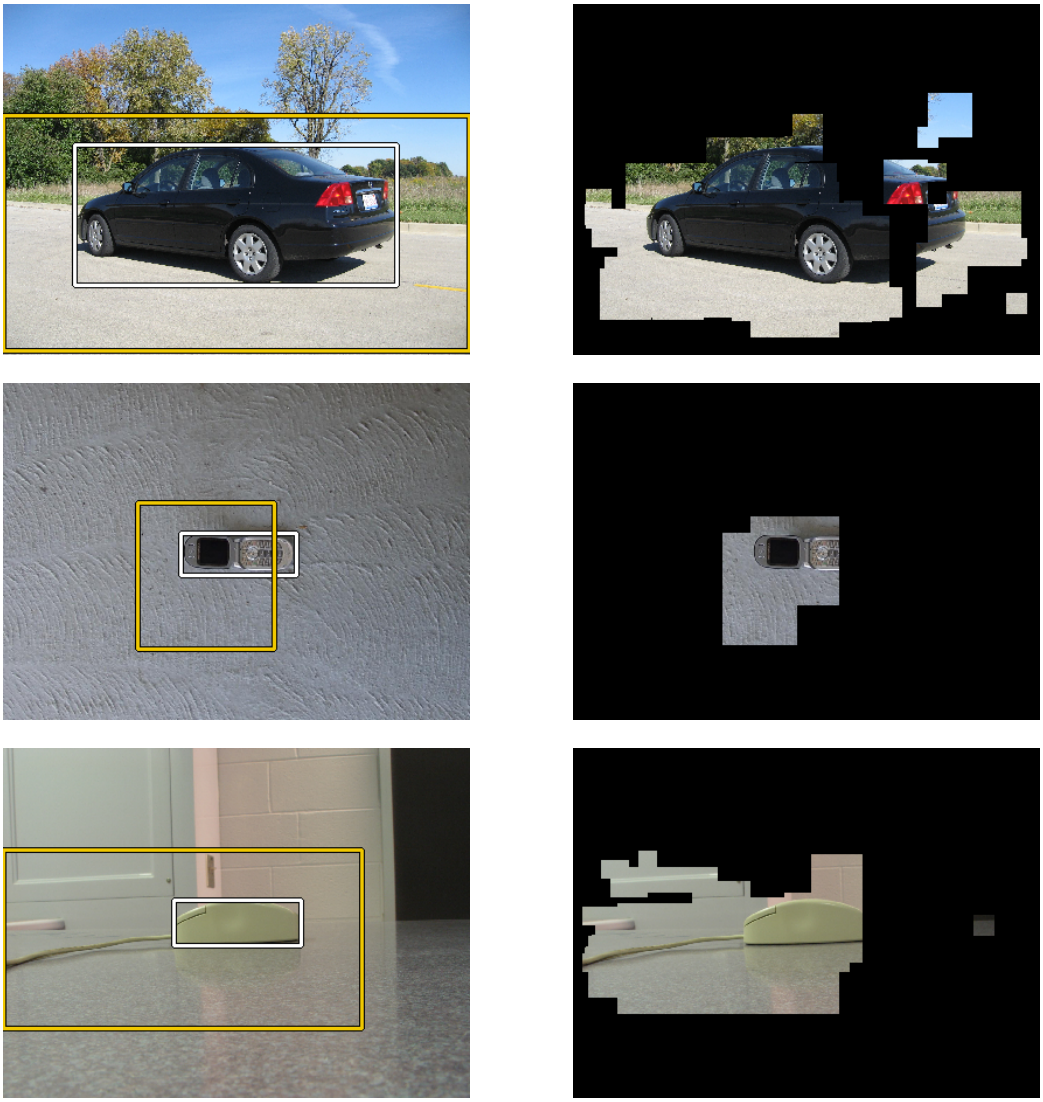


Fig. 7.9: Example results for the HOG model on six classes from 3D Object Categories dataset. Visualization is analogous to figure 7.4.

8. COMBINED MODEL

As the evaluations of the color-based model and the hog-based model suggest, both features are complementary since they model different visual properties. Therefore, we now combine both features into a single discriminative model which takes both models into account for each pixel location.

8.1 Color-and-Hog Model

Recall that our assumption is that the wanted objects are characterized by the features we use. Thus, we expect the objects to have both distinct colors and distinct gradient features. Obviously, these assumptions usually do not hold entirely, which is also indicated by our evaluations of the color model and the HOG model. For example, most logo classes have distinct colors and distinct gradients while flowers usually feature only weak visual HOG words.

We conclude that the best estimation for the location of the wanted object is where both models consistently find positive pixels. If only one model captures the actual object while the other model detects overly large areas, the combined result will be dominated by the better model. If, however, both features perform reasonably, the combined model will remove background areas where the individual models disagree.

We thus now determine locations, where positive pixels from both models are present by applying a logical AND-operator which combines color and gradient evidence at each pixel of image i. Thus our combined set of positive pixels $B(i)$ is the intersection of the positive pixel sets determined by the individual models:

$$B(i) = B_c(i) \cap B_{hog}(i) \tag{8.1}$$

Based on $B(i)$, we can determine bounding boxes as before.

Note that for the combined model, we perform an intersection of the individual models. This is equivalent to a logical AND operator for which

the "true" value is a neutral element. Therefore, pixels which are classified positive by a model do not affect the result unless they are also classified positive by the other model. In other words, a weak model which detects large false positive areas does not corrupt the result of the other model as long as both include the wanted object. In the extreme case, one model may detect every pixel of an image as positive without harming the result of the other model. Due to this behavior, we designed both models such that if they perform poorly, they tend to over detect the respective object.

The downside of this approach is that if both models find discriminative features which are found at different sub regions of the respective object, the method will fail. This happens for example if for a flower class, where HOG features are intuitively not suitable, a strong reference HOG feature is found since it is underrepresented in the negative set. Similarly, strong reference features may lead to the exclusion of "weaker" features which are also typical for the respective object class. This often leads to detecting sub regions of an object which are highly discriminative while removing less discriminative parts of the respective object.

It is also important to mention that if one feature F yields an empty set $B_F(i)$ we do not intersect both features which would lead to an empty set $B(i)$. In this case, we use the set $B_{F'}(i)$ of the respective other feature F'. In the extremely rare case of both features returning an empty set, we use the full image as bounding box as before (while evaluating the detection with 0 overlap).

8.2 Evaluation

For evaluation of the combined model, we again plot overlap-recall curves for all classes based on bounding boxes we obtain from the respective positive pixels.

8.2.1 Brand Logos

The results for the logo classes are shown in figure 8.1. The combined model outperforms both individual models for all logo classes, except arguably "Pepsi". For most classes, however, the improvement is considerable.

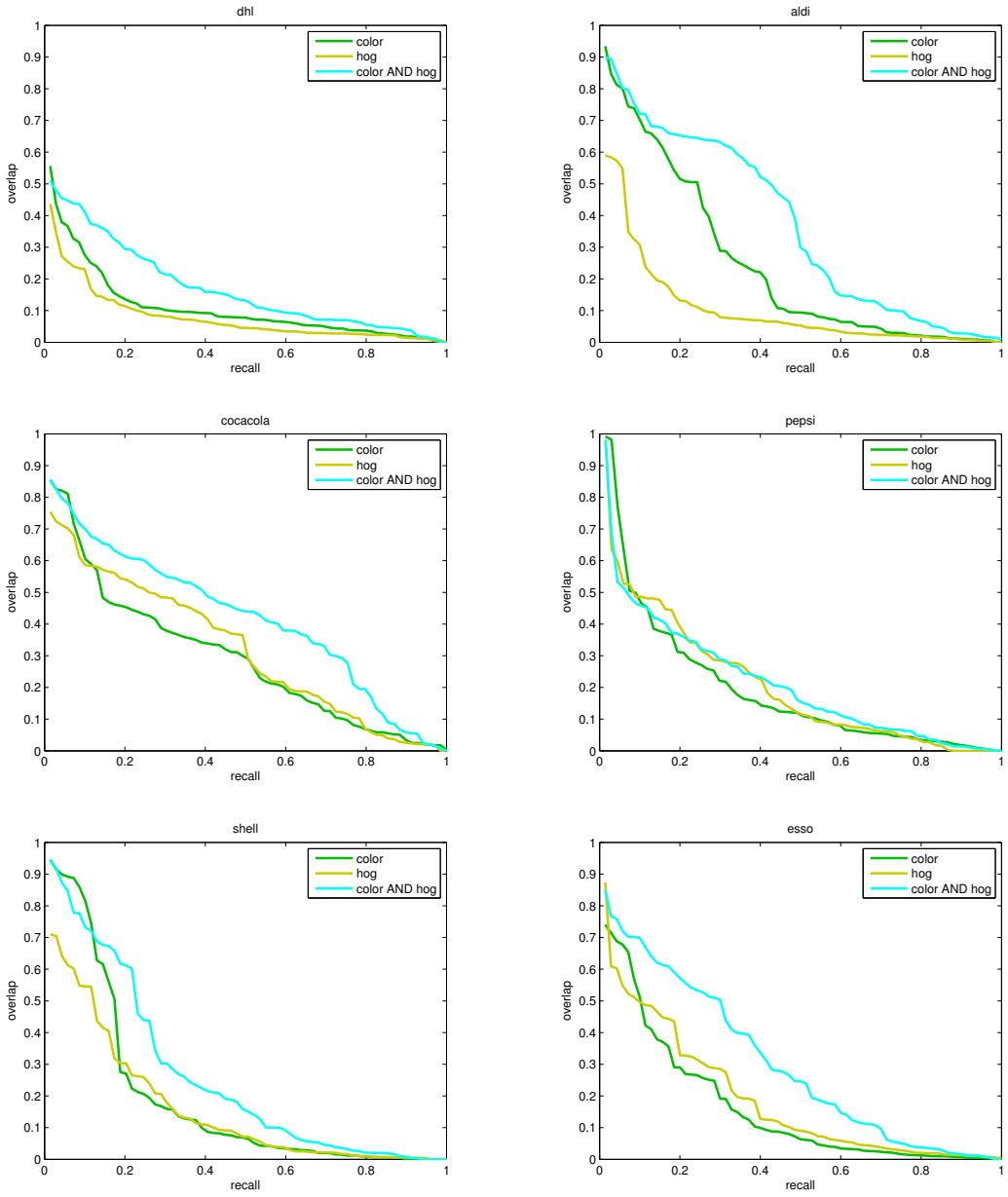


Fig. 8.1: Overlap-Recall curves for each class from FlickrLogos-6, based on combination of color and HOG features. For comparison, both features are also shown individually.

Fig. 8.2: Example results for three logo classes from FlickrLogos-6 for combined feature model. In the right image of each pair, the positive pixels $B_c(i)$ of the color model are shown as green pixels, the positive pixels $B_{hog}(i)$ based on the HOG model are blue and the combined model (i.e., the intersection of both) is visualized by cyan pixels. All three corresponding bounding boxes are shown in the left image of each pair whereas the green box is the color-based box, the yellow box is derived from HOG features and the cyan box is the final box from the combined model.

Fig. 8.3: Example results for three logo classes from FlickrLogos-6 for combined feature model. Visualization is analogous to figure 8.2.

We also show a few examples in figures 8.2 and 8.3. The left image in each row is the original image. Also, the resulting bounding box (cyan) of the combined model is shown along with the boxes we obtain from the color model (green) and HOG model (yellow) individually for comparison. The right image of each pair shows the positive pixels $B_c(i)$ of the color model (green), the positive pixels of the HOG model $B_{hog}(i)$ (blue), and the positive

pixels of the combined model $B(i)$ (cyan).

For all examples, the bounding box of the combined model is the best estimation except for the "Shell" example where the color model yields a slightly better bounding box. For this example, the color-based box already perfectly describes the object location. For the remaining examples, the individual models complement each other by removing different background areas and hence mutually correct false positive pixels. Thus, the examples illustrate the results indicated by the overlap-recall curves.

8.2.2 Flowers

For the flowers dataset, the results are shown in figure 8.4. Interestingly, while the combined model performs better than the HOG model, it does not improve the overall results beyond the color model. Since the curve is slightly lower, the HOG model even removes a few true positive pixels found by the color model in some cases. Recall, however, that we plot all classes combined into a single curve which means that a few outlier classes may significantly affect the curve.

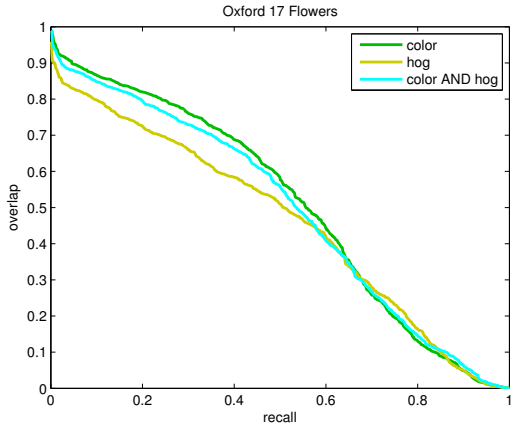


Fig. 8.4: Overlap-recall curve for Oxford 17 Flowers, based on combined models and on HOG and color for comparison.

In figure 8.5, three example results are shown. Again, we show all three bounding boxes and the sets of positive pixels for each model. For the first example, the HOG model and the color model complement each other, thus the combined model is better than the color model. The reason is that the

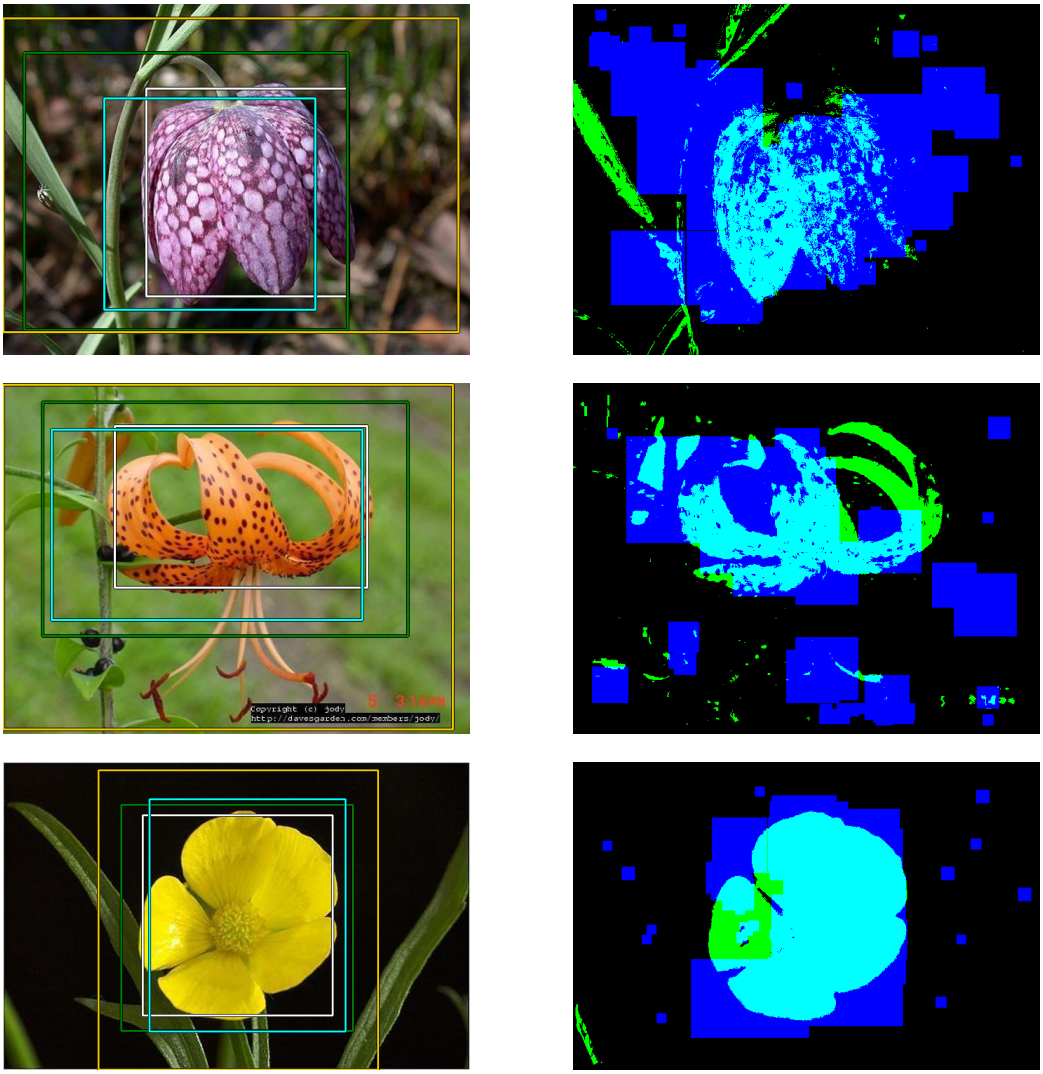


Fig. 8.5: Example results from Oxford 17 Flowers for combined feature model. Visualization is analogous to figure 8.2.

HOG model removes some of the false positive pixels caused by the false positive green pixels on the left. In the second example, the HOG model removes a few true positive pixels on the right side of the flower (which in this case does not lead to a combined bounding box which is inferior to the color box). For the third example, the color model already yields an almost

perfect bounding box and the inferior HOG model does not affect this result by its larger positive region. The combined bounding box is hence almost identical to the color-based bounding box.

8.2.3 3D Object Categories

For the 3D Object Categories dataset, the overlap recall curve of the combined model is shown in figure 8.6 along with the curves of both individual features. Combining the features yields superior results compared to the individual models which indicates that the two models are complementary to some degree, i.e., they produce different false positive pixels.

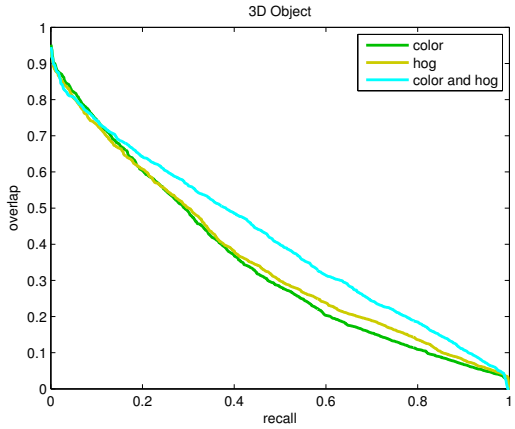


Fig. 8.6: Overlap-recall curve for color model, hog model, and combined model on 3D Object Categories dataset.

The qualitative results in figure 8.7 support this observation and show that the combined model yields the best bounding boxes for all examples. The second example, however, shows a situation where the color model fails and the HOG model dominates the result.

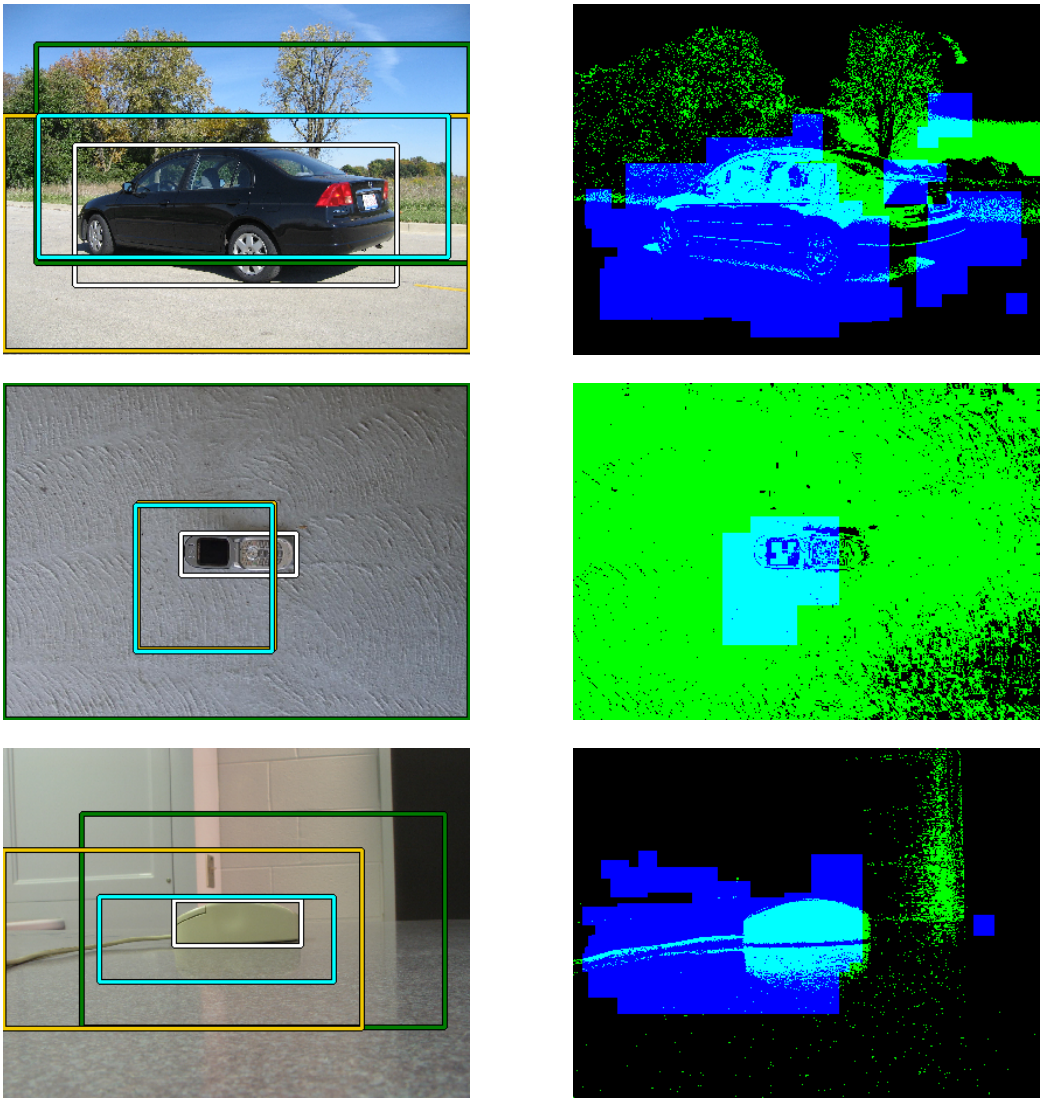


Fig. 8.7: Example results from 3D Object Categories for combined feature model. Visualization is analogous to figure 8.2.

9. MULTIPLE BOUNDING BOXES

In the previous experiments, we have only examined the problem of finding one single positive instance of a wanted object class within each positive image. This is appropriate in cases where only a single object instance is present per positive image. However, some positive images contain multiple instances of the wanted object class and thus we modify our approach in order to be able to find multiple object annotation in each positive image. Therefore, we now change our goal to finding all instances which are present within the positive images.

9.1 Heuristic for Multiple Bounding Boxes

As explained in section 5.1.2, our approach already allows creating multiple bounding boxes in one image i from the set of positive pixels $B(i)$. We simply use an EM algorithm [13] for estimating the overall distribution by a mixture of k Gaussians from $B(i)$ and create bounding boxes analogous to the single-Gaussian case.

In detail, for a given number of mixture components k, the EM algorithm returns a set of parameters $\Theta_k = \{\alpha_0, ..., \alpha_{k-1}, \theta_0, ..., \theta_{k-1}\}$ where the α_i are mixture weights and the $\theta_j = (\mu_j, \sigma_j^2)$ are the parameters of the respective Gaussian distribution. Thus, we obtain the following probability distribution over pixels $\mathbf{p}$:

$$p(\mathbf{p}|\Theta_k) = \sum_{j=0}^{k-1} \alpha_j p(\mathbf{p}|\theta_j) \tag{9.1}$$

where $p(\mathbf{p}|\theta_j)$ is a Gaussian density function with parameters μ_j and σ_j^2.

However, we do not know how many object instances are present in the respective image i. Thus, we try to determine if increasing k significantly helps explaining the spatial distribution of positive pixels $B(i)$. For this purpose, we compute the log-likelihood of parameter set Θ_k given the data

$B(i)$ for different values of k, i.e., for $k \in \{1, 2, 3, 4, 5\}$. The likelihood L of parameter set Θ_k given data $B(i)$ is defined by

$$L(\Theta_k|B(i)) = \prod_{\mathbf{p}\in B(i)} p(\mathbf{p}|\Theta_k) \tag{9.2}$$

The log-likelihood is then given by

$$\log(L(\Theta_k|B(i))) = \log\left(\prod_{\mathbf{p}\in B(i)} p(\mathbf{p}|\Theta_k)\right) = \sum_{\mathbf{p}\in B(i)} \log\left(\sum_{j=0}^{k-1} \alpha_j p(\mathbf{p}|\theta_j)\right) \tag{9.3}$$

Note that increasing k usually allows explaining the data better, i.e., the log-likelihood increases for larger k. Thus, simply selecting the value for k which yields the highest absolute log-likelihood is not desirable, because this would be the largest value for k. We need to determine if increasing k really leads to detecting several objects i.e., several separate positive pixel areas or if one single area is simply forced to be divided into k components. Fortunately, in the latter case, the difference in log-likelihood tends to be relatively small because if for instance we have a single object, the distribution $p(\mathbf{p}|\Theta_1)$ for $k = 1$ already explains the data accurately and increasing k only scarcely improves the log-likelihood. If we, however, actually have two separate objects in $B(i)$ which are at different locations of the image, increasing k significantly improves the log-likelihood.

We thus select the value k^* for $k \in \{1, ..., 5\}$ which yields the largest relative increase of log-likelihood. Let the relative increase $i(k)$ in log-likelihood for $k > 1$ be defined by

$$i(k) = 1 - \frac{\log(L(\Theta_k|B(i))}{\log(L(\Theta_{k-1}|B(i))}. \tag{9.4}$$

Then our choice for k^* is the number k of mixture components which increases the log-likelihood the most. However, since we want to introduce a slight bias towards larger k, we only require that $i(k) > 0.8 \cdot i(k-1)$ in order to prefer k over $k-1$

Also, we demand an increase of at least 0.025. If no $k > 1$ produces such increase, we keep $k^* = 1$ mixture components, since in this case increasing k does not considerably improve the mixture model with regards to how accurately it models the data. The value of 0.025 is again selected experimentally.

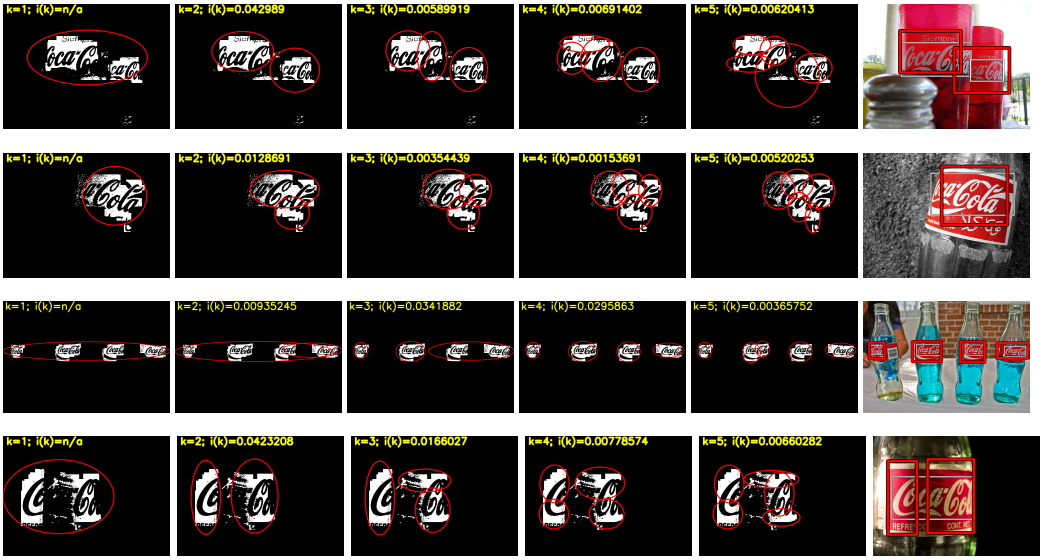


Fig. 9.1: Example results for our multi-bounding box heuristic.

Also note that this method of course does not take into account whether the found areas are actually different objects, false positive pixels, or one object which is "accidentally" separated into two areas by our discriminative model.

Figure 9.1 shows a few examples for our heuristic. Each row of images shows a binary map of positive pixels which have been returned by our initial model. We show the 2D Gaussian Mixture functions fitted into the positive pixels by the respective uncertainty ellipses, i.e., ellipses centered at μ_k with axis lengths of $\beta\sigma_x$ and $\beta\sigma_y$, respectively (see equation 5.2). In each image, the values of k and $i(k)$ are given in the upper left corner. The rightmost image in each row shows the resulting bounding boxes. The first example shows a situation where our heuristic correctly detects two objects. Note that the increase of likelihood is clearly largest for $k = 2$. In the second example, the threshold of 0.025 is never surpassed, so we keep $k = 1$. In the third example, the best increase is found for $k = 3$, however, $i(k = 4) > 0.8 \cdot k(k = 3)$, so this is an example where we prefer the larger $k = 4$ due to our bias factor. The final image illustrates a situation where our heuristic, despite behaving as expected, fails due to an incorrect set of positive pixels.

9.2 Additional Filters and Merging

Note that creating multiple bounding boxes introduces a few geometric issues. Since we cannot actually control the shape of our bounding boxes, we may obtain boxes with invalid aspect ratios. Thus, we define intuitive common sense thresholds on the aspect ratios of bounding boxes and remove boxes which violate them. For instance, we remove boxes where the ratio of long side divided by short side is more than 3.5.

Also, we may find boxes which are too small with regards to the size of the image or which do not actually contain a reasonable number of positive pixels due to an irregular or scattered shape of the underlying positive area. For all three kinds of degenerate bounding boxes, we again introduce intuitively reasonable thresholds.

For verification of these thresholds we test manually annotated bounding boxes in the experimental setup explained above on the Bikini dataset. We evaluate these thresholds and find that only 1% of the ground truth rectangles in the training set would be filtered out by our thresholds. Note that we do not violate our assumption of not requiring any object annotations for the desired object class, since this experiment merely verifies that our thresholds are reasonable. In fact, it does not depend on a specific object class, we only need to filter out boxes which are highly unlikely to describe any object.

Also, if we detect multiple rectangles that overlap more than 10%, we simply merge them into their common convex hull. Note that this does not affect multiple non-overlapping detections on the same object as for example in the bottom example of figure 9.1.

9.3 Hough Voting

After applying the aforementioned method, an instance of the desired object may still be broken up into multiple parts, i.e., we may obtain multiple bounding boxes for a single object as for example in the bottom row of figure 9.1. If these bounding boxes do not have enough overlap, they will not be merged during the post-processing explained in the previous section. For some classes of rigid objects, this happens regularly, since the respective objects consist of multiple indicative parts with gaps in between. In other words, for some classes, the initial model identifies positive pixels which form two disconnected areas. For example, consider the logos in figure 9.2,

where the red rectangles are the results of our bounding box estimations. Obviously, both logos are regularly split into two parts which have similar relative positions and sizes.

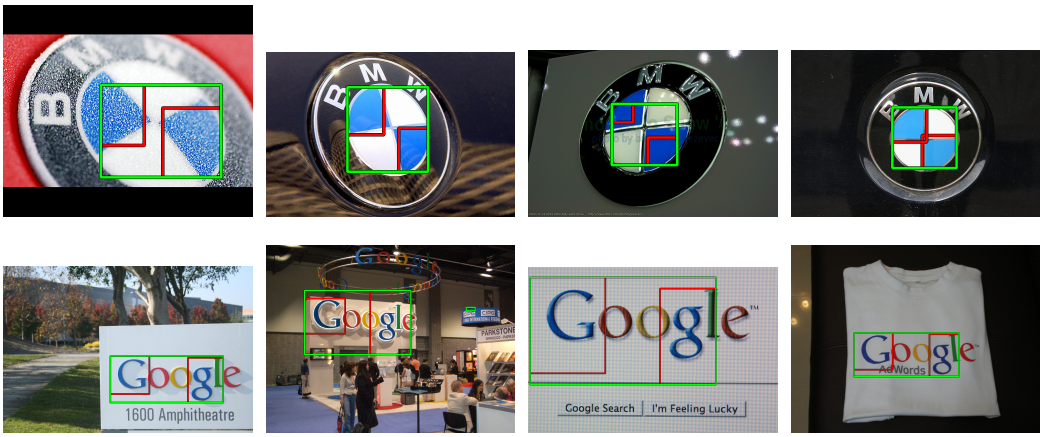


Fig. 9.2: Example results for the Hough voting algorithm.

We therefore try to determine if similar pairwise geometrical rectangle configurations are present in a large number of positive images of an object class and then merge the parts. By pairwise geometrical configuration we mean the relative positions of two rectangles and their aspect ratios.

Hough Voting is a method for spotting such regularities with acceptable computational effort. We first build a four-dimensional histogram over pairwise geometrical configurations of all pairs of rectangles from each positive image. The histogram dimensions reflect the signed horizontal and vertical distances of the upper left corners of the two rectangles, and both of their respective aspect ratios. These four values describe the relative pairwise position of a pair of rectangles and also their shapes.

Based on experiments, we decide to use 7 bins for each distance value and 4 bins for each aspect ratio. The bin widths are determined dynamically based on the maximum and minimum values among all rectangle pairs of the respective positive image set. Thus, before building the histogram we determine all pairwise configurations present in the positive images and set the respective bin sizes depending on the maximum and minimum values we observe.

Afterwards, all possible pairs of rectangles among all positive images are binned into our histogram. However, we again exclude extreme configurations

which are very unlikely to describe a single object. In detail, we exclude pairs where the absolute horizontal or vertical distance between both rectangles is more than five times the width or height, respectively, of any of the two rectangles, or where the area of one rectangle is more than three times larger than the other rectangle.

In figure 9.2 a few examples are shown. In each row, one particular histogram bin is visualized by four example images. The red pairs of rectangles have similar relative positions and similar respective aspect ratios, thus they fall in the same histogram bin.

After building the histogram, we find all bins, i.e., all pairwise rectangle configurations which occur in more than 10% of all images. Beginning with the largest bin, we then iteratively merge the respective rectangle pairs in their respective images into single rectangles. Also, if one of the merged rectangles is also part of other pairs of rectangles, these pairs are removed from the respective histogram bin in order to prevent merging the same rectangle with different corresponding rectangles.

In figure 9.2 the results of our method are shown as green rectangles which are obtained after merging the respective red pairs of rectangles. Note that only four examples are shown for each histogram bin, while we actually observe the respective configurations more often (in more than 10% of the images).

It should be noted that obviously our method only covers objects which are regularly split into two rectangles, which is sufficient for the datasets we use. However, it is straightforward to enhance it for configurations consisting of more rectangles by re-running the algorithm multiple times.

9.4 Evaluation on Multiple Instances

We now adjust our evaluation method to the new problem of finding all possible instances instead of finding one instance per image. Thus, we change our overlap-recall curve to represent the set of all instances (instead of all images) on the recall-axis. This changes the meaning of the recall-axis in comparison to the previous overlap-recall-curves, since it now represents the relative number of instances present in the dataset.

Note that with one detection per image, false detections (i.e., bounding boxes which are entirely in the background) were reflected by an overlap of 0 for the respective image. Now, we have multiple detections per image and

thus can fail to find an object within one image multiple times, i.e., detect rectangles which are false positives. Therefore, we now need to consider the total number of false detections in order to obtain meaningful results. In other words, if we ignore the number of false detections, the optimal model is simply a model which "detects" all possible rectangles.

We first define which detections count as false detections. First, each detection which does not have any overlap with any ground truth rectangle is considered a false detection. Second, we establish a one-to-one relation between detections and ground truth rectangles. Thus, if we produce overlap with the same ground truth rectangle by multiple detections, only the detection with the highest overlap score is counted. The remaining detections are counted as false detections (unless they are associated with a different ground truth rectangle for which they are the best detection). Also, each detection can only be associated with one ground truth rectangle. If one detection overlaps with multiple ground truth rectangles, only the one with the highest overlap score out of the set of ground truth rectangles to which no higher scoring detection is yet assigned, obtains the respective score. The remaining ground truth rectangles obtain overlap score 0.0 unless a different detection can be associated with them.

The overlap-recall plots in the remainder of this work will state the number of false positive detections per image in the legends next to the respective method as "avg. # FP". Note that the number given is the average absolute number of false positive detections per image and is not to be confused with a relative false positive rate, i.e., the number of false detections among all negative instances. The latter would be a much lower number due to the large number of possible negative rectangles.

Note that some classes contain images with an extremely high number of instances. These images would affect the recall-axis in an disproportionate manner causing the curves to "shrink to the left" which makes them hard to compare. Therefore, we ignore images with more than 10 instances for the multi bounding box evaluation in order to keep our plots expressive.

9.4.1 Brand Logos

Our results on the logo classes from FlickrLogos-6 are shown in figure 9.3. For comparison, we also show the results of the single bounding box approaches, i.e., the curves from figure 8.1.

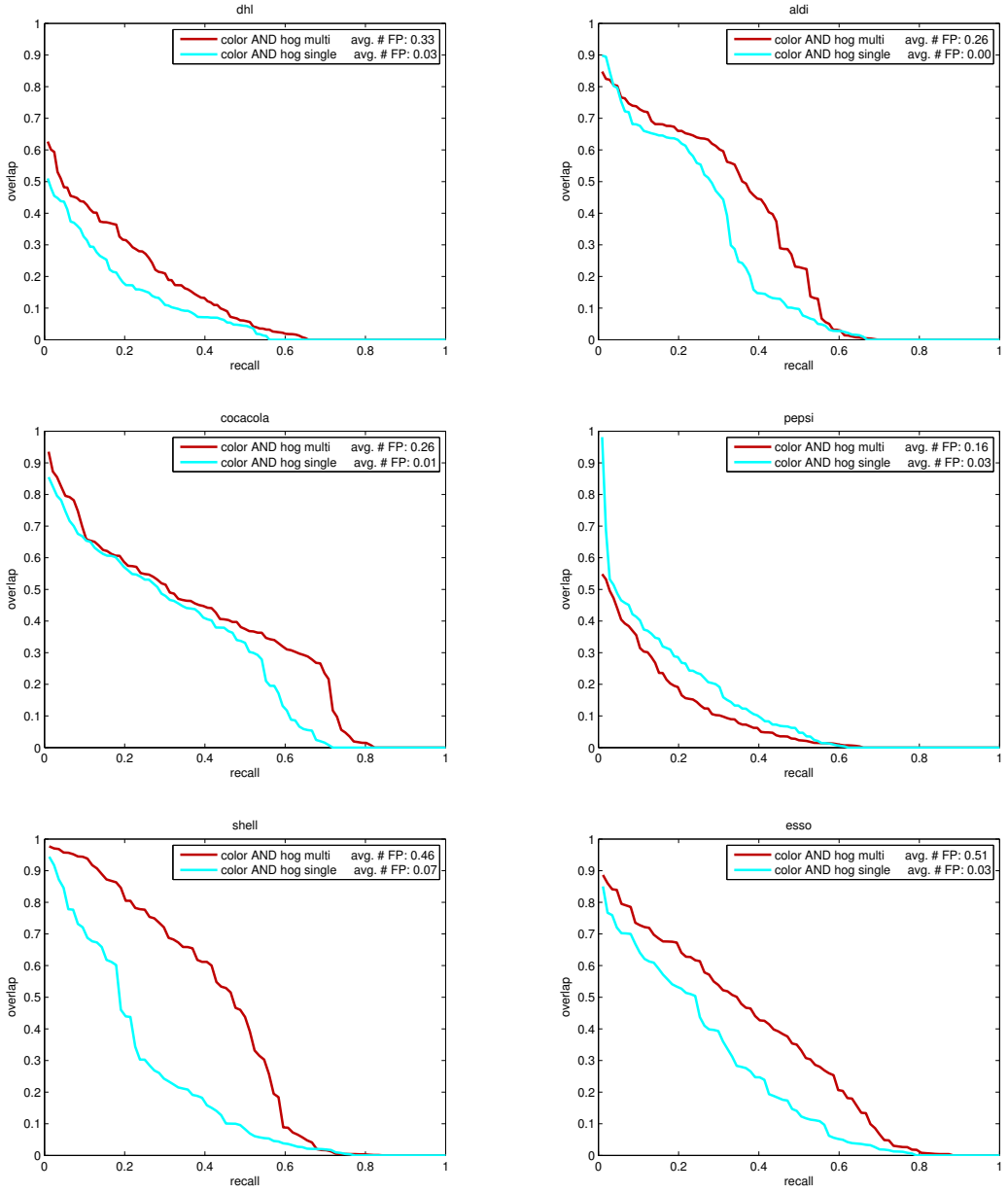


Fig. 9.3: Overlap-Recall curves for each class from FlickrLogos-6 based on combinations of colors and visual words with multi-bounding box heuristic (red) and with single detection per image (cyan) for comparison.

Note that one detection of the single method also represents the detection of one object instance which allows us to plot these results on our new recall-axis, since only the maximum number of detectable instances has changed.

The heuristic for multiple objects per image yields better curves for all classes albeit a higher number of false positives per image. The only exception is "Pepsi" where the heuristic does not improve the overlap-recall-curve.

We make similar observations for all remaining classes. The multi-box approach detects more objects, i.e., it drops to overlaps close to 0 later than the single-box approach. Also, the overlap is improved for five of our classes, i.e., the curves produced by the multi bounding box approach are above the curves for the single bounding boxes. However, the number of false detections is relatively large for some classes. The false detection rate ranges from 0.16 per image to 0.51 per image on average. Thus, we have to trade-off better detection results on the wanted objects against introducing false positive detections if we want to use multiple bounding boxes per image.

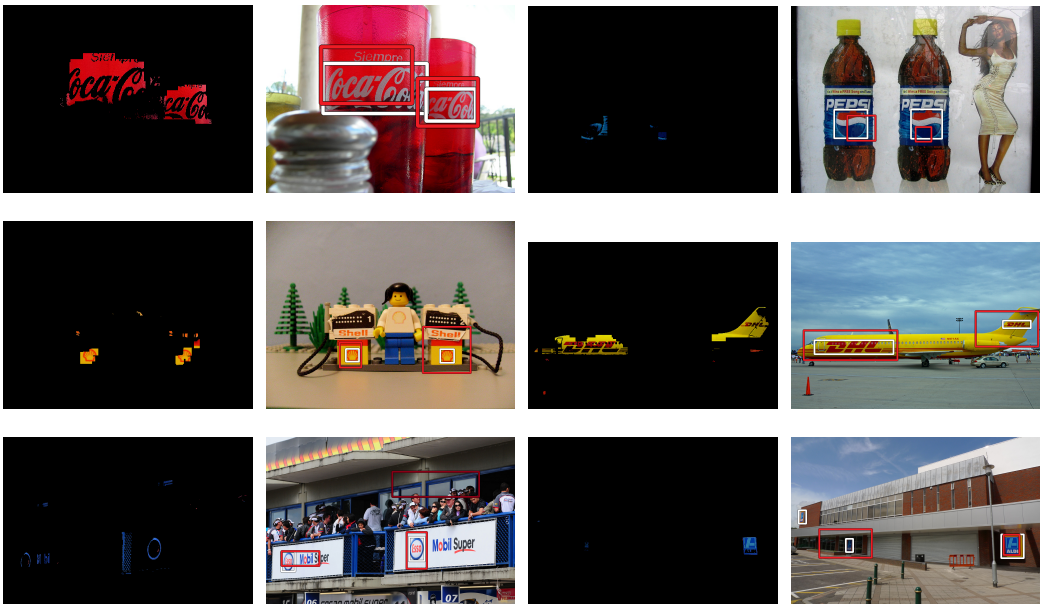


Fig. 9.4: Example results for multi bounding box detections on FlickrLogos-6. Each image pair shows the positive pixels of the combined model and the resulting bounding boxes as red rectangles. White rectangles are ground truth annotations.

In figure 9.4 a few example results of the multi bounding box detection are shown. For each image, positive pixels from the respective combined model are shown along with the resulting bounding boxes as red rectangles. In the first four examples, the multi-box heuristic works as intended. The lower row of images shows two common mistakes: The lower left example in the figure shows a false positive detection which is indicated by a dark red rectangle. In the lower right example, a large area of false positive pixels is included for one of the instances and one instance is missed.

9.4.2 Flowers

In the Oxford 17 Flowers dataset, there is a number of images showing multiple instances. However, recall that due to the annotation, a few instances may be labeled incorrectly. Overall, we still capture slightly more instances with the multi-object method as shown in figure 9.5 at the price of 0.44 false detections per image and slightly less total overlap. In figure 9.6, a few examples are shown where the multi-object heuristic yields good results.

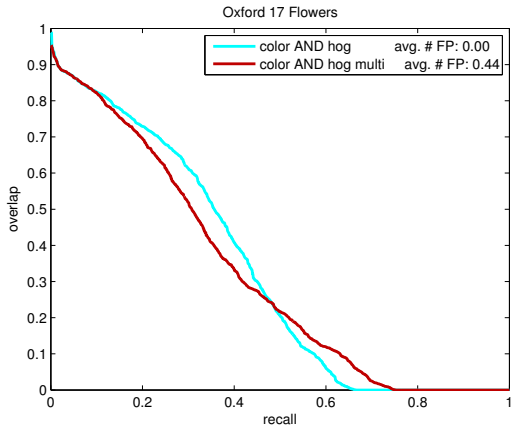


Fig. 9.5: Overlap-recall curves for combined model (cyan) and combined model with multi-bounding box detection (red).

9.4.3 3D Object Categories

For the 3D Object Categories the multi-object estimation cannot be expected to yield better results, since the dataset does not contain any images with

Fig. 9.6: Example results for multi bounding box detection on Oxford 17 flowers. Visualization is analogous to figure 9.4. Note that the upper right example is incorrectly labeled (three instances yield only one ground truth instance).

more than one object instance. It is thus not surprising that the overlap-recall curve obtained from the multi-object heuristic is not better than the single-object variant as shown in figure 9.7. The curve is even slightly worse, since the method fails in a few cases due to the reasons explained above, i.e., some objects are divided into multiple sub regions. Also, on average, we obtain one false detection in 49% of the images. In other words, if we do know that a dataset does not contain images with more than one object instance, the multi-bounding box heuristic (which is intentionally biased towards large numbers of instances) should not be applied.

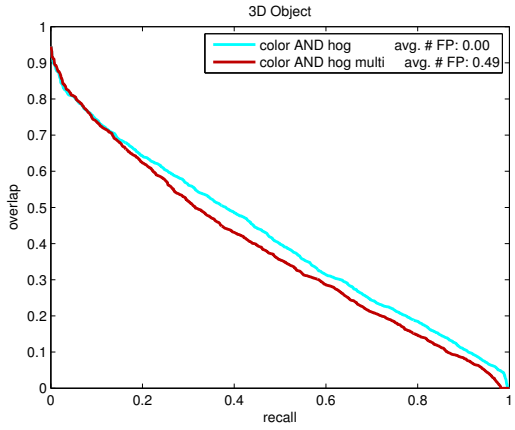


Fig. 9.7: Overlap-recall curves for combined model (cyan) and combined model with multi-object detection (red) on 3D Object Categories.

10. IMPROVING INITIAL BOUNDING BOXES BY LATENT STRUCTURAL SVM

The preceding chapters describe how we can find estimations for bounding boxes around objects based only on global image labels. We use heuristic decision rules based on feature statistics for color and gradient based features in order to find candidates for positive pixel locations which we enclose by rectangular bounding boxes. Our results show that our bounding boxes exclude a considerable amount of background. Therefore, we consider these bounding boxes a reasonable initial estimation.

In this chapter, we want to formulate the problem of improving the bounding box estimations in the positive images as a machine learning problem which can use our initial bounding boxes as starting points. Note, however, that not each bounding box estimation from our initial model necessarily describes a (different) object instance and that not every object instance is described by a bounding box estimation. Our goal is thus two-fold: First, we want to improve the initial estimations in terms of finding bounding boxes which are closer to the ground truth. Second, we want to remove surplus estimations. The case of finding additional instances, i.e., increasing the number of estimations in an image is not considered, since we have no further evidence supporting additional bounding box initializations.

The advantage of the machine learning approach is that it constructs a consistent model over co-occurrences of HOG features among all object instances of all positive images. The initial model on the other hand only relies on statistics which treat all features independently. Also, it performs an optimum rectangle search as opposed to heuristically fitting a rectangle into positive pixels. Another motivation for this approach is the work by Felzenszwalb et al. [20] where the authors suggest that object instances can iteratively be improved by treating properties of the ground truth annotations as latent variables. A thorough discussion on the motivation and implementation of the Latent Structural learning algorithm can be found at the end of this chapter in section 10.7.

For clarity, we denote instances and estimations by a single index i and do not explicitly add another index indicating the respective positive image in which instance i is located. Thus, in the remainder of this work, multiple positive instances or estimations may implicitly live in the same positive image.

Since we are searching for rectangles in our positive images, the output of our learning problem can be considered a *structured output label* [45], which is a multi-dimensional output label in contrast to the one-dimensional label y of a non-structured classifier. This output label renders our problem a structured learning problem. A structured learning problem is also characterized by an (almost) non-enumerable set of negative examples. Note that we also have such a negative set considering that the number of possible rectangles in the negative images is huge.

The second important aspect of our learning problem is that our positive training examples are noisy, since our initial estimations are created by heuristic models and often deviate considerably from the ground truth. In other words, the actual bounding box required for a machine learning problem is unknown. If we have such unknown and unobservable properties among our training examples, we can model these properties by *latent variables*.

For obtaining a model, we thus use a latent and structured version of the widely used Support Vector Machine (SVM) classifier, which allows updating our noisy training data and also mining useful negative examples.

Note that finding the best values for the latent variables during training corresponds to our goal of finding the optimal bounding boxes. This is a notable difference from conventional learning problems which aim at building a model for classifying or detecting instances in unknown data. Thus, while we implicitly also train a model, our goal is not to build an optimal classifier for "new" instances.

In the following section, we will first give a concise overview of the linear Support Vector Machine. Then we explain how a linear SVM can be used for structured problems and how we can model our problem as a latent learning problem. Finally, we devise a learning algorithm which iteratively improves our bounding boxes based on the latent structured problem formulation.

10.1 Linear SVM

Linear Support Vector Machines are a long-established and thoroughly researched implementation of linear decision functions. The following explanation of linear SVM follows descriptions found in [11], [46], [6], and [39].

In general, a linear decision function $f(x) : \mathbb{R}^d \to \mathbb{R}$ can be written as the scalar product of an adaptive parameter vector $\mathbf{w}$ (also called *model vector*) and some d-dimensional example vector x with an added bias parameter b:

$$f(x) = \langle \mathbf{w}, x \rangle + b \tag{10.1}$$

The example vector x here denotes a feature representation of a training example which is in our case a BOW histogram as explained below in section 10.5.1. Note that x may be extracted from a full image or a fixed rectangle. In this formulation, however, the region for feature extraction is immutable.

If positive and negative examples of the given instance space are linearly separable, we can find a vector $\mathbf{w}$ and bias parameter b such that for all positive examples x' we obtain $f(x') > 0$, and for all negative examples x'' we have $f(x'') \leq 0$. In fact, $\mathbf{w}$ and b then define a d-dimensional hyperplane which is located in between the positive and the negative data points, and the perpendicular distance of feature point x to the hyperplane is given by $f(x)\|\mathbf{w}\|^{-1}$. Thus, the sign of $f(x)$ corresponds to the class of example x (positive or negative), and the value $f(x) \in \mathbb{R}$, which is the distance to the hyperplane, can be interpreted as the *score* of example x.

10.1.1 Hard Margin SVM

We first explain the hard margin SVM variant. The term "hard margin" indicates that no training example may violate the hyperplane defined by the decision function we seek. In other words, each example must lie on the correct side of the hyperplane. We hence implicitly assume linearly separable training data for the moment, otherwise a hard margin decision function does not exist.

One way to obtain an optimal model vector $\mathbf{w}$ which yields such classification results is training a linear support vector machine on a set of training examples of the form (x_i, y_i) where x_i is the actual example instance and $y_i \in \{-1, 1\}$ is the known *label* of example i, i.e., the sign indicating the class of the training example (negative or positive). For a linear SVM, the

optimal decision hyperplane is created such that it provides a maximum *margin*. The margin is the orthogonal distance between the hyperplane and the closest data point x_k of the training set. If we thus seek a maximum margin, we have to find a vector $\mathbf{w}$ and bias b such that they maximize this distance:

$$\underset{\mathbf{w},b}{argmax} \left(y_k(\langle \mathbf{w}, x_k \rangle + b) \|\mathbf{w}\|^{-1} \right) \tag{10.2}$$

Note that we can scale $\mathbf{w}$ and b by a constant factor such that the unnormalized distance of the hyperplane to the closest data point x_k will be 1 and thus all remaining data points must have a distance of at least 1. This leads us to the following constraints:

$$\forall i \in \{1, ..., n+m\} : y_i(\langle \mathbf{w}, x_i \rangle + b) \geq 1 \tag{10.3}$$

If we want to determine the optimal (maximum) margin we thus must find the $\mathbf{w}$ and b which maximize $\|\mathbf{w}\|^{-1}$ and hence minimize $\|\mathbf{w}\|$ due to equation 10.2, while the constraints defined by equation 10.3 hold. Thus, the constrained quadratic optimization problem is finding

$$\mathbf{w}^*, b^* = \underset{\mathbf{w},b}{argmin} \frac{1}{2} \|\mathbf{w}\|^2 \tag{10.4}$$

$$s.t. \;\; \forall i \in \{1, ..., n+m\} : y_i(\langle \mathbf{w}, x_i \rangle + b) \geq 1 \tag{10.5}$$

The function to be minimized in equation 10.4 is called the *objective function*. Note that the minimization problem considers both $\mathbf{w}$ and b. Since the constraints must hold, the bias term b is implicitly optimized and does not appear in the optimization term 10.4. It can be shown (see [11]) that the optimal solution to this problem can be written as a linear combination of the n positive and m negative input training vectors with coefficients $\alpha_1, ..., \alpha_{n+m}$:

$$\mathbf{w} = \sum_{i=1}^{n+m} \alpha_i y_i x_i \tag{10.6}$$

The coefficients α_i are Lagrange multipliers which stem from solving the optimization problem. Constrained quadratic optimization problems like the

one at hand can be solved by differentiating a Lagrangian function $L(\mathbf{w}, b, \alpha)$:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^2 - \sum_{i=1}^{n+m} \alpha_i \left(y_i(\langle \mathbf{w}, x_i \rangle + b) - 1\right) \tag{10.7}$$

Finding the saddle point of $L(\mathbf{w}, b, \alpha)$ with respect to $\mathbf{w}$ (i.e., setting the derivative of $L(\mathbf{w}, b, \alpha)$ for $\mathbf{w}$ to 0 and solving for $\mathbf{w}$) yields the above solution in equation 10.6. Intuitively, the Lagrange multipliers can be interpreted as weights of the individual training examples. Determining the value of b is explained below.

A linear SVM is hence a decision function as defined in equation 10.1 where the model vector $\mathbf{w}$ is a weighted sum of the feature representations of all training examples. Substituting 10.6 in 10.1, we obtain the final linear SVM decision function

$$f(x) = \sum_{i=1}^{n+m} \alpha_i y_i x_i^T x + b \tag{10.8}$$

A more detailed explanation of the derivation of equation 10.8 from equations 10.4 and 10.5 can be found in appendix A. For the optimal solution of the maximum margin problem, the weight of all training examples which do not lie on the margin in feature space will obtain a weight of $\alpha_i = 0$. The remaining training examples (on the margin) which obtain a weight $a_i \neq 0$ are called *support vectors*. Since all support vectors lie on the margin, we can now set the bias b to the average of the scalar products of $\mathbf{w}$ with all support vectors x_j:

$$b = -\frac{1}{n_S} \sum_{j=1}^{n_S} (\langle \mathbf{w}, x_j \rangle - y_j) \tag{10.9}$$

where n_S is the number of support vectors. Note that in the hard margin case, a single support vector would be sufficient to compute b. However, in the soft-margin case explained below, we allow support vectors to violate the margin and even to lie on the "wrong" side of the hyperplane. As a consequence we need a numerically stable estimation for the offset which is the mean over all support vectors. Thus, the bias term b intuitively is an offset which shifts the result of the scalar product between $\mathbf{w}$ and an input feature vector. Otherwise, the margin would be supposed to lie in the origin of the feature space.

Since the linear SVM only involves computing a scalar product, i.e., a linear function, it can be evaluated efficiently. Also, the decision function can be re-arranged as explained below in order to implement efficient search strategies. Another advantage of linear SVMs is that they are relatively robust against over-fitting the training data, since only training examples which are selected as support vectors influence the decision function. This is an important property for our application, since our training data may be highly noisy.

10.1.2 Soft Margin SVM

Note that the optimization problem explained above assumes that the training examples are linearly separable. Real-world data, however, is often not linearly separable. Thus, the exact constraints for the optimization problem given in equation 10.3 which do not allow any misclassifications (each example must be on the correct side of the decision hyperplane) need to be adjusted. Recall that the constraints intuitively require that every training example be further away from the hyperplane than the margin, i.e., its classification score must be at least 1. For allowing misclassifications among the training examples, *slack variables* are introduced to the constraints which allow some deviation from this minimum target distance which yields a modified set of constraints:

$$\forall i \in \{1, ..., n+m\} : y_i(\langle \mathbf{w}, x_i \rangle + b) \geq 1 - \xi_i \tag{10.10}$$

where $\xi_i \geq 0$ is the slack variable. The resulting margin is referred to as *soft margin* [11], since it now allows misclassified training examples.

Note that with $\xi_i = 0$, we obtain the original constraint for example i, i.e., training examples with $\xi_i = 0$ are classified correctly. The remaining examples with $\xi_i \neq 0$ are either within the margin (if $\xi_i \leq 1$) or even on the wrong side of the decision hyperplane (if $\xi_i > 1$). Thus, the magnitude of the slack variables intuitively corresponds to the degree of violation of the margin. Hence the optimization problem must obviously penalize large slack values. This can be achieved by adjusting the original minimization problem in equations 10.4 and 10.5 as follows:

$$\mathbf{w}^*, b^* = \underset{\mathbf{w},b,\xi}{argmin} \left(\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n+m} \xi_i \right) \tag{10.11}$$

$$s.t. \;\; \forall i \in \{1, ..., n+m\} : y_i(\langle \mathbf{w}, x_i \rangle + b) \geq 1 - \xi_i \tag{10.12}$$

The parameter C is used to control the influence of the penalty induced by the slack variables. Intuitively, the larger C the closer the optimization problem is to the original formulation without slacks, since margin violations produce a larger penalty. If C is a small value, we implicitly allow more margin violations. Thus, C enables us to trade-off performance on the training data by minimizing the training error against model complexity and generalization capabilities of the trained model. In practice it is often difficult to predict a reasonable value for C and it is often selected from a large interval on a separate validation set or by cross validation.

The optimization problem is analogous to the problem for the hard margin SVM. The corresponding Lagrange function is

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^2 + C \sum_{i=1}^{n+m} \xi_i - \sum_{i=1}^{n+m} \alpha_i \left(y_i(\langle \mathbf{w}, x_i \rangle + b) - 1 + \xi_i \right) \tag{10.13}$$

Now the minimization task also involves minimizing with regards to the slack variables. The model vector then again becomes

$$\mathbf{w} = \sum_{i=1}^{n+m} \alpha_i y_i x_i \tag{10.14}$$

with $\alpha_i > 0$ for support vectors (active constraints) and $\alpha_i = 0$ for inactive constraints. How the Lagrange multipliers are derived is again illustrated in appendix A.

10.2 Linear SVM for Structured Output

Note that the above description of the linear SVM considers a two-class problem where each training example i has a binary label $y_i \in \{1, -1\}$ and the feature representation x_i of example i is immutable. A *structured* formulation of a learning problem [44], however, allows that the feature representation

$\Psi(x_i, y_i)$ of example i also depends on the label y_i. Therefore, a structured learning problem may involve a complex output label, i.e., a rectangle $y_i = (p_{x,i}, p_{y,i}, w_i, h_i)$ with upper left point $(p_{x,i}, p_{y,i})$, width w_i, and height h_i.

The major advantage of switching to a structured formulation of a learning problem concerns the selection of examples during training. Since our problem is characterized by a (nearly) non-enumerable number of negative examples (all rectangles in all negative images), we are faced with the problem of choosing useful examples for training our linear SVM. This problem can be addressed by devising a bootstrapping mechanism which iteratively mines the negative set for hard (i.e., useful) examples. As will be discussed below, this search for training examples can be directly included in the training algorithm when using a structured formulation, since the structured output label y_i corresponds to the position of the respective example in rectangle space Y among which the examples are distributed and which is hence to be searched for useful examples.

Specifically, in contrast to a decision function $f(x) : \mathbb{R}^d \rightarrow \mathbb{R}$ as described above, we now consider a function $f(x) : \mathbb{R}^d \rightarrow \mathbb{R}^e$ which maps a d-dimensional input example x_i to a structured, multivariate e-dimensional output label y_i which in our case is a rectangle.

The second major formal difference is that feature representation $\Psi(x_i, y_i)$ of example x_i now also depends on the label y_i. In other words, we shift the annotated rectangle of image i from the representation x_i to the label y_i. Thus, x_i now stands for all features found in image i and the subset of features used is defined by the label y_i. For example, in our problem y_i is the rectangle for which we need to extract a BOW histogram in image i. For different y_i we obviously obtain different histograms from the same x_i. In contrast, in the non-structured case, x_i is a fixed BOW histogram from the features of image i (either for the full image or for an invariable annotated rectangle).

It is important to mention that we only predict positive instances, i.e., each predicted label is supposed to describe an instance of the positive class. Specifically, for our problem this means that if we predict a rectangle, we implicitly predict that the rectangle includes an instance of the wanted object. If we want to express that our decision function predicts that the wanted object is not present in image i, we define that our prediction $\hat{y}_i$ is empty and hence write $\hat{y}_i = \emptyset$. Analogously, if i is a negative image, we also define its label to be an empty label and write $y_i = \emptyset$. For an empty label, the function $\Psi(x_i, \emptyset)$ is defined to return a zero vector $\overrightarrow{\mathbf{0}}$ of the same length as

a non-empty $\Psi(x_i, y_i)$. The reason for these definitions are elaborated below where we explain how our constraints are constructed. How (non-empty) feature vectors of negative rectangles are still included in the training process is also explained below.

10.2.1 Formulation of Structured Problem

In this section, we formulate the optimization problem for a structural SVM. We derive the optimization problem following [45] and [22]. As for the non-structured case, we begin with a hard-margin formulation.

First we restate the non-structured soft-margin optimization problem:

$$\mathbf{w}^*, b^* = \underset{\mathbf{w},b,\xi}{argmin} \left(\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n+m} \xi_i \right) \tag{10.15}$$

$$s.t. \;\; \forall i \in \{1, ..., n+m\} : y_i(\langle \mathbf{w}, x_i \rangle + b) \geq 1 - \xi_i \tag{10.16}$$

In the structured case, we again want to minimize the L2-norm of the model vector plus the sum over all slack variables as in equation 10.15 whereas we now have a more general set of constraints as explained below.

Note that for a structured problem the error on a training example does not have to be binary as for the standard binary SVM formulation. In practice, the error between a prediction $\hat{y}$ and a label y is usually quantified by an error function called *loss* function $\Delta(\hat{y}, y)$. The loss function has the following requirements: $\Delta(\hat{y}, y) \geq 0$ for $\hat{y} \neq y$ and $\Delta(\hat{y}, y) = 0$, i.e., the correct prediction must have minimum loss 0.

As in [45] we define our constraints in terms of this loss function, beginning with the hard-margin formulation:

$$\forall i \in \{1, ..., n+m\} : \forall \hat{y} \in Y \backslash y_i : (\langle \mathbf{w}, \Psi(x_i, y_i) - \Psi(x_i, \hat{y}) \rangle) \geq \Delta(\hat{y}, y) \tag{10.17}$$

The bias term b of equation 10.3 is missing in equation 10.17 since (under the definitions for empty detections given in the previous section), it can be integrated into the scalar product by simply adding one entry to the feature vector, namely b, and setting the corresponding entry in $\mathbf{w}$ to 1. Also, instead of determining it as explained in equation 10.9, one can set the additional entry of the feature vector to a constant value and let the learning algorithm determine an appropriate weight for b as suggested in [40] (which,

however, changes the primal optimization problem as also discussed in [40]). We therefore omit b in the following explanations.

Intuitively, the constraints of equation 10.17 indicate that the scalar product of model vector $\mathbf{w}$ and the difference vector between the feature representation for label y_i and prediction $\hat{y}$ must be at least the value of the loss caused by prediction $\hat{y}$. For instance, let i be a positive example. If prediction $\hat{y}$ is equal to label y_i, i.e., a perfect prediction, the scalar product will be 0, since the difference vector will be $\overrightarrow{\mathbf{0}}$. Thus, for the perfect prediction the constraint will become $0 \geq 0$ which reflects that example i is already predicted correctly under the current model vector $\mathbf{w}$.

The other extreme case is predicting that the best label for positive example i would be an empty label $\hat{y}_i = \emptyset$ as explained in the previous section. In this case, the scalar product will be $\langle \mathbf{w}, \Psi(x_i, y_i) \rangle$, since $\Psi(x_i, \hat{y}_i = \emptyset) = \overrightarrow{\mathbf{0}}$. The resulting constraint requires that the score of example i must be larger or equal to the loss $\Delta(\emptyset, y)$ which in a reasonable scenario would be 1 for a fully incorrect prediction. In other words, the constraint for the example which was misclassified will require that the respective feature representation gains a score of at least 1 under model vector $\mathbf{w}$.

Given an appropriate loss definition, all other possible predictions $\hat{y}_i$ are in between the two extreme cases. Note that the aforementioned examples illustrate that, for binary prediction, i.e., if we can only predict the correct label y_i or the empty label, our set of constraints is equivalent to the constraints of a non-structural SVM as defined in equation 10.3. Note that since $\Psi(x_i, y_i) - \Psi(x_i, \hat{y})$ has a positive sign for an empty prediction in a positive example, and a negative sign in the opposite case, the multiplication with the label (i.e., -1 or $+1$) of the binary SVM in equation 10.3 is also included in equation 10.17. Thus, the structured formulation can be viewed as a generalization of the binary case if a negative prediction is modeled by the empty prediction. This can be verified by considering all possible combinations of labels and binary loss values of either 0 or 1.

To be more specific, if x_i is a positive example, and the prediction of the model vector is correct, i.,e., $\hat{y} = y_i$, the constraint is satisfied and we obtain the trivial constraint $\langle \mathbf{w}, \overrightarrow{\mathbf{0}} \rangle \geq \Delta(y_i, y_i) = 0$. If, however, for the positive example an empty detection is predicted (i.e., $\hat{y} = \emptyset$), and we define a loss of $\Delta(\hat{y} = \emptyset, y_i \neq \emptyset) = 1$ for this case, we obtain a constraint of the following form:

$$\langle \mathbf{w}, \Psi(x_i, y_i) - \overrightarrow{\mathbf{0}} \rangle \geq \Delta(\hat{y} = \emptyset, y_i \neq \emptyset)$$
$$\langle \mathbf{w}, \Psi(x_i, y_i) \rangle \geq 1$$

which is equivalent to the non-structured counterpart in equation 10.5 except for the missing bias term omitted for the reasons explained above. For negative training examples, we can make analogous observations. A correct empty detection for a negative example x_i again yields a trivial constraint $\langle \mathbf{w}, \overrightarrow{\mathbf{0}} \rangle \geq \Delta(y_i, y_i) = 0$. An incorrect detection, i.e., $\hat{y} \neq \emptyset$ yields the following constraint if we again assume $\Delta(\hat{y} \neq \emptyset, y_i = \emptyset) = 1$ for this case:

$$\langle \mathbf{w}, \overrightarrow{\mathbf{0}} - \Psi(x_i, \hat{y}) \rangle \geq \Delta(\hat{y} \neq \emptyset, y_i = \emptyset)$$
$$- \langle \mathbf{w}, \Psi(x_i, \hat{y}) \rangle \geq 1$$

which indicates that the false positive detection $\hat{y} \neq \emptyset$ must be assigned a score of ≤ 1 (note the negative sign in contrast to the above constraint for the positive example). Overall, our constraints thus require that positive examples obtain a score ≥ 1 while negative examples must obtain a score ≤ 1 which is equivalent to the non-structured constraints in equation 10.4 (besides the omitted b).

For the soft-margin variant, we add slack variables which allow violating our constraints analogous to the non-structural SVM. As defined in equation 10.11, the slacks also have to be included in the minimization problem. The full optimization problem for the structured soft-margin SVM is then:

$$\mathbf{w}^* = \underset{\mathbf{w},\xi}{argmin} \left(\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n+m} \xi_i \right) \tag{10.18}$$

$$s.t. \;\; \forall i \in \{1, ..., n+m\} : \forall \hat{y} \in Y \setminus y_i : (\langle \mathbf{w}, \Psi(x_i, y_i) - \Psi(x_i, \hat{y}) \rangle) + \xi_i \geq \Delta(\hat{y}, y_i) \tag{10.19}$$

Note that the structural SVM formulation assumes that our labels y_i are the true bounding boxes of the wanted objects in the respective positive images. Since our rectangles are however provided by our initial model described in the previous chapters, we cannot make this assumption. Therefore, we define our problem as a latent learning problem as explained in the following section.

10.3 Adding latent variables

A learning problem which involves properties of training examples which are unknown but relevant for the result of the training, can be considered a *latent* learning problem. More specifically, we have a latent problem, if the training examples have some properties which cannot be observed directly and can be described by a variable which is then called a *latent variable*.

10.3.1 Prediction With Latent Variables

Since we do not know if the rectangles we obtain from our initialization approach actually describe the locations of wanted objects, the correct rectangle position can be considered latent. In other words, we do not know the actual best rectangle position and we cannot directly observe it, since we do not have manual annotations. Overall, the instances X of our learning problem have thus the following form:

$$X = \{(x_1, y_1, h_1), ..., (x_{n+m}, y_{n+m}, h_{n+m})\} \tag{10.20}$$

where x_i is our i-th training instance and the rectangle position for example i is now modeled by latent variable h_i. The variable y_i is again the label of the example. Our first estimate for h_i is therefore the rectangle detected by our initial statistical model. Also, our feature representation $\Psi(x_i, y_i, h_i)$ now depends on the latent variable. The negative examples have latent variables only for consistency since we obviously know that the best rectangle for a negative example is empty.

Analogous to [53], the fact that the rectangle position is now modeled in h_i renders the label y_i a binary label which may either be positive or negative, i.e., $y_i \in \{-1, 1\}$ representing negative and positive examples, respectively. The structured property of our problem is now only modeled in the latent variable which arguably contradicts our definition of a structured problem given above due to the binary output label y_i. Adapting the terminology of [53], however, we still call our problem a latent structured problem.

As explained above for the structured problem, negative predictions and negative examples are defined to return empty feature vectors, i.e., we have $\Psi(x, y, h) = \overrightarrow{\mathbf{0}}$ if $y = -1$.

The optimization problem including latent variables can now be formu-

lated as determining a prediction function $f_{\mathbf{w}}(x)$ of the following form

$$f(x_i) = \underset{(y_i,h_i)\in Y\times H}{argmax} \langle \mathbf{w}, \Psi(x_i, y_i, h_i)\rangle \tag{10.21}$$

The prediction function thus now returns not only the most likely label y_i for example i but the most likely combination of label and value for the latent variable h_i. Thus, a prediction is now denoted by $(\hat{y}, \hat{h})$.

Training this prediction function hence involves simultaneously finding an optimal model vector $\mathbf{w}$ and an optimal configuration for the latent variables. Our final goal is the latter, since we are interested in the locations of the desired objects.

10.3.2 Regularized Risk for Structural SVM

For the derivation of the latent problem as a difference of two convex functions[1] which is required for the algorithm we want to use, we must formulate the optimization problem in terms of minimizing the training error while searching for the maximum margin. Following [51], for non-latent structural SVM learning problems, the minimization problem can be stated in terms of the *regularized risk* $R_\Delta(f)$:

$$R_\Delta(f) = \|\mathbf{w}\|^2 + C\sum_{i=1}^{n+m}\left(\underset{\hat{y}\in Y}{max}[\Delta(\hat{y}, y_i) + \langle \mathbf{w}, \Psi(x_i, \hat{y})\rangle] - \langle \mathbf{w}, \Psi(x_i, y_i)\rangle\right) \tag{10.22}$$

The regularized risk quantifies the accumulated training error over all training examples expressed as the difference between expected error (loss) and score of the difference of the feature vector $\Psi(x_i, y_i)$ under the actual example label y_i and the feature vector $\Psi(x_i, \hat{y})$ under the prediction $\hat{y}$ by model vector $\mathbf{w}$. It also takes into account the size of the margin of the hyperplane defined by $\mathbf{w}$.

Note that minimizing the regularized risk is equivalent to the primal structured optimization problem stated in equations 10.18 and 10.19. This can be verified by first solving 10.19 for ξ_i:

$$\forall i \in \{1, ..., n+m\} : \forall \hat{y} \in Y \setminus y_i : \xi_i \geq \Delta(\hat{y}, y_i) - \langle \mathbf{w}, \Psi(x_i, y_i) - \Psi(x_i, \hat{y})\rangle \tag{10.23}$$

[1] $f(x)$ is *convex* if $\forall x_1, x_2 : \forall \lambda \in [0, 1] : f(\lambda x_1 + (1-\lambda)x_2)) \leq \lambda f(x_1) + (1-\lambda)f(x_2)$

which yields

$$\forall i \in \{1, ..., n+m\} : \forall \hat{y} \in Y \setminus y_i : \xi_i \geq \Delta(\hat{y}, y_i) - \langle \mathbf{w}, \Psi(x_i, y_i) \rangle + \langle \mathbf{w}, \Psi(x_i, \hat{y}) \rangle \tag{10.24}$$

Note that the upper bound for the right side of this inequation is reached for the one prediction $\hat{y}$ which maximizes the expression. Also note that the feature vector for the true label y_i is constant and does not depend on $\hat{y}$. Thus, overall we obtain for each ξ_i

$$\xi_i = \underset{\hat{y} \in Y}{max}[\Delta(\hat{y}, y_i) + \langle \mathbf{w}, \Psi(x_i, \hat{y}) \rangle] - \langle \mathbf{w}, \Psi(x_i, y_i) \rangle \tag{10.25}$$

If we substitute this in the objective function of the primal problem in equation 10.18 and skip the constant factor $\frac{1}{2}$ for readability, we obtain the risk minimization problem:

$$\underset{\mathbf{w}}{min} \left(\|\mathbf{w}\|^2 + C \sum_{i=1}^{n+m} \left(\underset{\hat{y} \in Y}{max}[\Delta(\hat{y}, y_i) + \langle \mathbf{w}, \Psi(x_i, \hat{y}) \rangle] - \langle \mathbf{w}, \Psi(x_i, y_i) \rangle \right) \right) \tag{10.26}$$

which is equivalent to minimizing $R_\Delta(f)$ as defined in equation 10.22.

10.3.3 Optimization Problem for Latent Variables

As already mentioned above, in the latent case our feature representation $\Psi(x_i, y_i, h_i)$ now involves latent variables. Our loss function $\Delta(\hat{y}_i, \hat{h}_i, y_i, h_i)$ again quantifies the training error of prediction $(\hat{y}_i, \hat{h}_i)$ by means of the desired prediction function $f(x_i)$ for example i. Now the loss in general may also depend on the latent variable and the prediction for the latent variable. The derivation closely following [51], however, requires that the loss function does not depend on the latent variable h_i, i.e.,

$$\Delta(\hat{y}_i, \hat{h}_i, y_i, h_i) = \Delta(\hat{y}_i, \hat{h}_i, y_i) \tag{10.27}$$

Since the loss-function is typically not convex, it is replaced by a piecewise linear convex upper bound as shown in [45]. As shown in [51], the optimization problem of minimizing the regularized risk for the latent case can then be written as a minimization task for $n + m$ training instances:

$$\min_{\mathbf{w}} \left(f(\mathbf{w}) - g(\mathbf{w}) \right) \tag{10.28}$$

where

$$f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\mathcal{L}(\mathbf{w}) \tag{10.29}$$

with

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{n+m} \max_{(\hat{y},\hat{h}) \in Y \times H} \left(\langle \mathbf{w}, \Psi(x_i, \hat{y}, \hat{h}) \rangle + \Delta(\hat{y}, \hat{h}, y_i) \right) \tag{10.30}$$

and

$$g(\mathbf{w}) = C \sum_{i=1}^{n+m} \max_{h \in H} \langle \mathbf{w}, \Psi(x_i, y_i, h) \rangle \tag{10.31}$$

In (10.29) and (10.31) C is the margin parameter as explained above. For our implementation we experimentally choose $C = 10^3$.

Note that equations 10.28 to 10.31 describe the full optimization problem for the structured case with latent variables, since the constraints are implicitly included in the optimization problem. Since the optimization problem in (10.28) is the difference of two convex functions (which is not convex), we can solve it using an instance of the CCCP algorithm [52] as described in the next section.

10.4 Training algorithm

Our training algorithm is a modified version of the Convex-Concave Procedure (CCCP) algorithm [52]. The CCCP algorithm is designed for latent structural SVM training and thus solves the optimization problem defined in equation 10.28. Our version of the CCCP algorithm mostly follows the implementation by Yu and Joachims [51]. The following explanation also borrows from [50]. Algorithm 1 is a pseudo-code formulation of the CCCP algorithm.

The input to the algorithm are our positive and negative training examples. Note that each negative training example refers to one negative image, while multiple positive examples may stem from the same image. For clarity, we again do not explicitly denote the image by an extra index. Also, we

Input
Positive and negative examples $(x_1, y_1), ..., (x_{n+m}, y_{n+m})$;
Initial rectangle estimations $\hat{r}_i$ for positive examples

1 $\forall i \in 1, ..., n : h_i^* = h_i = \hat{r}_i$; $\mathbf{w}_0 = \mathbf{0}$; $t = 0$;
2 **repeat**
`// Construct hyperplane` $\mathbf{v}_t$
3 $\mathbf{v}_t = \sum_{i=1}^{n+m} \Psi(x_i, y_i, h_i^*)$;
`// Solve optimization problem`
4 $\mathbf{w}_{t+1} = \underset{\mathbf{w}}{argmin}[f(\mathbf{w}) + \langle \mathbf{w}, \mathbf{v}_t \rangle]$;
`// Infer latent variables` h_i^* `for each`
`// positive example` $x_i \in X$
5 $h_i^* = \underset{h \in H_i}{argmax} \langle \mathbf{w}_{t+1}, \Psi(x_i, y_i, h) \rangle$;
6 $t = t + 1$;
until $(f(\mathbf{w}_{t+1}) - g(\mathbf{w}_{t+1})) - (f(\mathbf{w}_t) - g(\mathbf{w}_t)) < \epsilon$;

Output
Latent variable h_i^* for each positive example

Algorithm 1: CCCP training algorithm

provide one initial rectangle estimation $\hat{r}_i$ for each positive example from our initial statistical model.

The main idea of the algorithm is to iteratively train an SVM model and update the latent variables in an alternating manner. The CCCP algorithm can thus be viewed as an Expectation Maximization algorithm.

In each iteration t of the algorithm, a model vector $\mathbf{w}_{t+1}$ is trained in line 4 by optimizing the problem stated in equation 10.28. Since this problem is the sum of a convex and a concave function and hence not convex, equation 10.28 is replaced by

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}}{argmin}[f(\mathbf{w}) + \langle \mathbf{w}, \mathbf{v}_t \rangle] \tag{10.32}$$

Thus, the function to be minimized becomes convex. This can be done, since $\mathbf{v}_t$ is a hyperplane which upper bounds $-g(\mathbf{w})$ for the current $\mathbf{w}_t$. The hyperplane $\mathbf{v}_t$ is chosen such that

$$\forall \mathbf{w} : f(\mathbf{w}) - g(\mathbf{w}) \leq f(\mathbf{w}) - g(\mathbf{w}_t) + \langle (\mathbf{w} - \mathbf{w}_t), \mathbf{v}_t \rangle. \tag{10.33}$$

Thus, $g(\mathbf{w})$ is approximated by a linear Taylor approximation $-g(\mathbf{w}_t) + \langle(\mathbf{w} - \mathbf{w}_t), \mathbf{v}_t\rangle$ at point $\mathbf{w}_t$ and overall, the right side of inequation 10.33 is hence a global upper bound on the objective function $f(\mathbf{w}) - g(\mathbf{w})$ for the current model vector $\mathbf{w}_t$. Therefore it describes a linear bound to the objective function in the current optimum $\mathbf{w}_t$ yielding a linear tangent in point $\mathbf{w}_t$. The linear hyperplane $\mathbf{v}_t$, i.e., the derivative of $g(\mathbf{w}_t)$ is then

$$\mathbf{v}_t = \sum_{i=1}^{n+m} \Psi(x_i, y_i, h_i^*) \tag{10.34}$$

Since the constant negative value of function $g(\mathbf{w}_t)$ at point $\mathbf{w}_t$ is not relevant for the minimization task, the upper bound minimization problem in equation 10.33 can be reduced to the term in equation 10.32.

By solving the optimization problem, an SVM model is trained under the assumption that the current h_i^* is the optimal setting for h_i. The actual optimization problem in line 4 is then solved using the cutting plane algorithm [22] which is implemented as explained below.

In line 5 new estimations h_i^* for the latent variables h_i are determined from the given instance-dependent search spaces H_i. The new estimation is the one h which maximizes the SVM score, i.e., the scalar product with the current model vector $\mathbf{w}$, of the respective example's feature representation $\Psi(x_i, y_i, h)$.

Recall that the goal of the CCCP algorithm is to minimize the difference $(f(\mathbf{w}_t) - g(\mathbf{w}_t))$ according to equation 10.28. Therefore it terminates if this difference does not change by more than a given threshold ϵ.

Algorithm 1 is a general formulation of the algorithm which requires a number of problem-specific implementations which are explained in the following section.

10.5 Implementation

In this section we describe our implementation decisions for the CCCP training algorithm. It is mostly based on [51] and [53], but there are a few differences and practical decisions which are explained in the remainder of this section.

10.5.1 Example Representation and Latent Variables

For a latent structured learning problem, our examples need to have the form (x_i, y_i, h_i) and a feature representation $\Psi(x_i, y_i, h_i)$. In our scenario, we use a HOG-based example description. As explained in section 7.1, HOG features model gradient distributions by building histograms over the respective gradient orientations. In many recent, successful object detection approaches [12, 20, 30, 47, 53], rectangular image regions are described by concatenating HOG cells. This representation requires that relative cell locations be invariant among all positive examples. In other words, a HOG cell at a certain relative location within given rectangles in multiple images must model the same part of the respective object. This strategy naturally requires that the rectangles are relatively accurate.

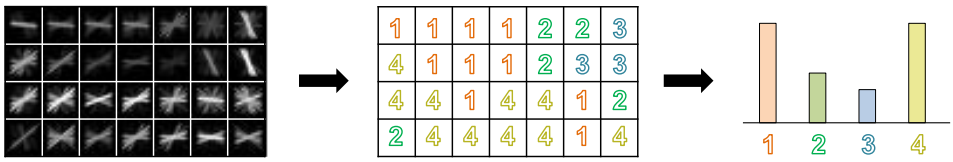


Fig. 10.1: Schematic visualization of the Bag-of-Words paradigm. Features, e.g. HOG cells, are extracted (left), cluster ids are assigned (middle) and a occurrence histogram is built (right).

Since our estimated rectangles are often inaccurate, this strategy is not appropriate for our scenario. We need a feature representation which is less strict and where the location of certain HOG cells within our rectangle may vary among different rectangles. A representation which fulfills these requirements is provided by the Bag-of-Visual-Words paradigm explained in section 7.2. Since each HOG feature is assigned a cluster id, we can simply count the occurrence frequency of each cluster id within a given rectangle which yields a visual word histogram. For comparability, we L1-normalize all histograms such that they sum to 1.0.

BOW histograms dismiss the relative location at which a certain feature has been observed and are thus invariant against spatial permutations of the same set of cluster ids. Also, if we assume that object instances always yield identical numbers of the same cluster ids, bag-of-word histograms for rectangles which include the respective object along with some background will only differ in the occurrence frequencies of their respective background words.

Therefore, BOW histograms are to some degree robust against this type of noise. On the downside, BOW histograms encode less information about the underlying image region due to clustering errors and ignoring spatial information. Figure 10.1 visualizes the process of building a BOW histogram.

Overall, in our scenario, x_i is thus a multi-scale grid of visual words based on HOG features as visualized in figure 10.2. The label y_i describes if the example is positive or negative (which is known from the global image label) and $h_i = (x_{h_i}, y_{h_i}, w_{h_i}, h_{h_i}, s_{h_i})$ models the unknown actual optimal bounding box for the respective desired object on the dense HOG cell grid. It also models the scale s_{h_i} on which the respective rectangle is defined. Our function $\Psi(x_i, y_i, h_i)$ thus builds a BOW histogram for rectangle h_i using the cluster ids of all cells within the rectangle given by h_i on scale s_{h_i} for positive labels y_i. Note that in the first iteration of our training algorithm, the latent variable is initialized by a rectangle estimation from our initial statistical model which lives on the pixel grid of the image. We thus need to map this rectangle to the HOG grid which is explained in detail in section 10.5.4 below.

Recall that intuitively, each predicted rectangle $\hat{h}$ is interpreted as a positive prediction. Therefore, negative predictions, i.e., $\hat{y} = -1$ are modeled as empty predictions, i.e., $\Psi(x_i, \hat{y}, h_i) = \overrightarrow{\mathbf{0}}$. Analogously, the feature vector of a negative example with $y_i = -1$ is also empty $\Psi(x_i, y_i, h_i) = \overrightarrow{\mathbf{0}}$. Note that during the training process explained below, we still obtain constraints for negative feature vectors due to the formulation of the constraints.

Since our initial model may return false positive rectangles $\hat{r}_i$, we may include some instances for which h_i hence is not meaningful. During our algorithm, the value of h_i may still converge to an actual desired object which is only partially included in the initial estimation $\hat{r}_i$.

It should also be mentioned that for efficiency, all images are scaled to a maximum side length of 800 pixels prior to feature extraction.

10.5.2 Loss Function and Model Vector Learning

Following [53], we use a binary loss function which is defined as follows:

$$\Delta(\hat{y}_i, \hat{h}_i, y_i, h_i) = \begin{cases} 0 & \text{if } y_i = \hat{y}_i \\ 1 & \text{otherwise} \end{cases} \tag{10.35}$$

This definition fulfills the aforementioned requirement in equation 10.27 as it does not depend on h_i. For a positive example, the loss is 1.0 if the prediction is negative due to the empty prediction having a higher score than the actual feature representation of example i. If the prediction is positive for a positive example, the loss is 0. For negative example the loss yields the opposite respective values.

In line 4 of algorithm 1, the optimization problem is solved, i.e., an SVM model vector $\mathbf{w}$ is computed for the current iteration. Recall that the function which is minimized in line 4 is an upper bound on the objective function of the latent structured problem. Here, we implicitly assume our latent variables as given and train the SVM model analogous to a non-latent structural SVM. Thus, we now train a structural SVM by solving the constrained optimization problem for structural SVM as defined above in section 10.2.1 with the only difference that h_i is used as the structured label (instead of y_i as in section 10.2.1). For this training, we use an implementation of the cutting plane algorithm by Joachims et al. [22].

Since the algorithm is explained in detail in [22], and our implementation is based on their implementation, we only give an overview of the most important aspects and the practical considerations for our scenario. In general, the algorithm works by iteratively constructing sets of constraints in the form of 10.19. As mentioned above, the difference to the structured constraints is that in the latent case, the feature representation $\Psi(x_i, y_i, h_i)$ depends on h_i instead of y_i.

On negative examples, the constraints are determined by finding the highest scoring rectangle within a negative image. Usually, these constraints are referred to as the "most violated constraints". Since all other rectangles in the respective negative images yield lower scores, the most violated constraints are intuitively the most useful negative examples. Note that the respective most violated constraint will have loss 1, since any prediction except the empty prediction with label $\hat{y} = -1$ in a negative image is incorrect.

For illustration, let i be a negative instance. Note that we only search for one most violated constraint per negative image, so index i also denotes a unique negative image in contrast to positive examples. Now let $\hat{h}_i$ be the predicted highest scoring rectangle in i. Since we predict a rectangle other than the empty rectangle, the label of our prediction is positive, i.e., $\hat{y}_i = 1$. Hence the feature representation of the prediction is also not empty. The actual feature representation of the negative example i, however, is $\Psi(x_i, y_i, h_i) = \overrightarrow{\mathbf{0}}$ as defined above.

Overall, the constraint for image i and predicted rectangle $\hat{h}_i$ will then evaluate to

$$(\langle \mathbf{w}, \overrightarrow{\mathbf{0}} - \Psi(x_i, \hat{y}_i, \hat{h}_i) \rangle) + \xi_i \geq 1 \tag{10.36}$$

or

$$(\langle \mathbf{w}, \Psi(x_i, \hat{y}, \hat{h} \rangle) \leq -1 + \xi_i \tag{10.37}$$

which indicates that the respective negative example defined by predicted rectangle $\hat{h}$ must obtain a score of less than -1.0 plus slack value. Thus, the highest scoring rectangle in each negative example will yield a soft constraint which requires that the respective feature vector must have a score below -1.0. In other words, the cutting plane algorithm (and thus the structured formulation of the problem) provides a built-in hard negative mining in each iteration. This is a major advantage over standard binary SVM training.

It is easy to see that if no example in a negative image obtains a score better than the empty detection (which has score 0, since the empty detection is represented by an empty vector), the respective constraint will evaluate to

$$(\langle \mathbf{w}, \overrightarrow{\mathbf{0}} - \overrightarrow{\mathbf{0}} \rangle) + \xi_i \geq 0. \tag{10.38}$$

Note that the loss is 0 in this case since our prediction is correct. This constraint evaluates to the trivial constraint $0 + \xi_i \geq 0$ which indicates that example i has already been "learned" correctly, since it has been predicted correctly. Then, for the current iteration, negative image i will not contribute any (useful and hence active) constraints since no rectangle was found in the respective negative image which violates the requirement formulated in 10.38. Otherwise it would have had a higher score than the empty prediction and be used for a constraint as in equation 10.37.

Constraints are obtained from positive examples in the same way as for negative examples. We again determine the constraint for the highest scoring prediction for a positive example i. For positive examples, we know (or assume in the latent case) that rectangle h_i is the correct rectangle. In other words, the highest scoring prediction for a positive example may either be $\hat{y}_i = y_i$ and implicitly $\hat{h}_i = h_i$ or the empty detection for which we know that it is incorrect. In detail, if the empty detection has the highest score, we obtain the following constraint for positive example i:

$$(\langle \mathbf{w}, \Psi(x_i, y_i, h_i) - \overrightarrow{\mathbf{0}} \rangle) + \xi_i \geq 1 \tag{10.39}$$

which indicates that example i, which was incorrectly predicted, contributes a constraint of the form

$$(\langle \mathbf{w}, \Psi(x_i, y_i, h_i) \rangle) \geq 1 - \xi_i \tag{10.40}$$

This constraint requires that the feature vector obtained from rectangle h_i must have a score of at least $1.0 - \xi_i$. For the correct prediction, example i again only contributes a trivial constraint, since the respective loss value will be 0 as well as the difference in the scalar product.

Together with equation 10.11, the constraints form a quadratic optimization problem which is solved for $\mathbf{w}$ using the method of [21]. Given the new model vector, the next iteration is carried out beginning with determining new constraints. If the constraints do not change anymore, the algorithm terminates. The final model vector is a solution of the structured optimization problem stated in line 4 of algorithm 1.

Note that for the first iteration, we set $\mathbf{w} = \overrightarrow{\mathbf{0}}$ which leads to incorrect predictions on positive examples (no example has a higher score than the empty detection) and thus yields constraints of the form 10.39. For negative examples, all rectangles will have the same score, so the first rectangle encountered in our search for the most violated constraint is chosen for the most violated constraint. Thus, in the first iteration, the set of negative constraints consists of "random" rectangles, if we assume that the content of the negative images is not biased towards certain scenes. Also recall that in the following iterations we explicitly search for hard negatives.

10.5.3 Efficient Implementation of Rectangle Search

As mentioned above, the optimization problem requires searching for the highest scoring predictions, i.e., rectangles, among the negative images multiple times in each CCCP iteration since each run of the cutting plane algorithm is iterative itself. Also, in line 5 the positive images are searched for the highest scoring rectangles form given search spaces H_i. Therefore, the search for the highest scoring rectangle in an image is a bottleneck of the algorithm as it is executed multiple times for each image in each iteration.

For this reason, we use an efficient search algorithm analogous to the approach by Lampert et al. [24] which can be used for a linear decision function, i.e., a linear SVM. In practice, the following search is done over up to 8 neighboring scales (both larger and smaller scales). For readability, we explain it for one given scale.

First, let w_k be the k-th element of model vector $\mathbf{w}$. Also, let x_k be the unnormalized k-th element of some feature vector $\Psi(x, y, h)$, i.e., the absolute occurrence frequency of visual word k in rectangle h. Then the dot product computing the score of the example is

$$\langle \mathbf{w}, \Psi(x, y, h) \rangle = n^{-1} \sum_k w_k x_k \tag{10.41}$$

where n is the number of visual words, i.e., HOG cells, in rectangle h. Each value w_k is a constant weight for the k-th visual word which is summed up x_k times for the scalar product. In other words, w_k can be interpreted as the contribution of a single occurrence of visual word k. If we now associate this contribution w_k to each cell c_k of the HOG grid which is assigned cluster id k, we obtain a cell contribution $w_{c_k} = w_k$ for each cell c_k of the image.

The scalar product of equation 10.41 can therefore be re-written as

$$\langle \mathbf{w}, \Psi(x, y, h) \rangle = n^{-1} \sum_{c_k \in h} w_{c_k} \tag{10.42}$$

where $c_k \in h$ denotes that cell c_k lies within rectangle h.

Such sums over constant weights within rectangles can be efficiently computed by integral images which reduce equation 10.42 to four look-ups in a pre-computed table. Using this method, we can efficiently evaluate each rectangle in a given image. It is worth mentioning that in [24], a branch-and-bound algorithm is described which can be used to further speed up the maximum search.

10.5.4 Search Space for Inference of Latent Variables

Our initial statistical model yields rectangles $\hat{r}_i$ defined at pixel level whereas our latent variables need to live on the multi-scale HOG grid due to our feature representation. This section explains how we initially select a rectangle h_i on the HOG grid for a given estimated rectangle $\hat{r}_i$ and describes how we restrict the number of possible aspect ratios during the inference of latent variables.

In line 1 of algorithm 1, we initialize the latent variables h_i by the initial rectangle $\hat{r}_i$. In practice, this implicitly involves mapping the pixel-based $\hat{r}_i$ to a HOG-grid-based rectangle h_i, since we use h_i to extract our BOW histograms as explained in the previous section. We therefore first define the

Fig. 10.2: Schematic visualization of the multi-scale Bag-of-Words grid for an image (left). The multi-scale image pyramid is visualized by a few scales in the center of the figure whereas the HOG cells (visualized by their centers) are projected into the image in original size. On the right, the selected scale is shown and the HOG cells are highlighted which are used for representing the rectangle from the left image.

following set R of reasonable aspect ratios (or templates) in terms of pairs of vertical and horizontal numbers of hog cells:

$$R = \bigcup_{k=7}^{10} ((k, \lfloor 100/k \rfloor) \cup (\lfloor 100/k \rfloor, k)) \tag{10.43}$$

Intuitively, we thus use aspect ratios ranging from $(7, 15)$ to $(15, 7)$ whereas all templates have roughly the same area of 100 HOG cells. We now choose the one rectangle on the multi-scale HOG grid and from our set of aspect ratios R which has the maximum overlap with $\hat{r}_i$ as our initial h_i. This procedure is visualized in figure 10.2 where on a rectangle $\hat{r}_i$ on pixel-level (left image) is mapped to a rectangle h_i on the HOG grid (right image).

Note that, as mentioned before, we search over multiple scales, i.e., we include up to 8 neighboring scales of the current estimation in both directions, i.e., larger and smaller scales, limited by the minimum and maximum scales available from HOG extraction.

During our search, we also restrict the set of valid aspect ratios in order to prevent rectangle detections which simply "shrink" or take on invalid aspect ratios. For illustration, consider line 5 of algorithm 1. The latent variable

h_i will be selected such that the scalar product of the current model vector and the BOW histogram of rectangle h_i becomes maximally large. Each HOG cell within rectangle h_i will contribute either a negative or a positive summand to the scalar product. If we do not impose any constraints on the shape of h_i, we enable the algorithm to choose extremely small or, with regards to aspect ratio, arbitrarily shaped rectangles around HOG cells with large contributions. Small rectangles in particular will be preferred over large rectangles which include some background or noise introducing negative summands to the scalar product.

In this work, we do not tackle the problems of occlusion and extreme deformations. Therefore we can further reduce the search space H_i for the latent variables as we know that we are searching for instances of the same object in each positive image. In other words, we expect the aspect ratio of the desired object to be roughly the same in each image. For this reason, the height and width of each latent variable, i.e., w_{h_i} and h_{h_i} become independent from the respective instance i in our scenario. Therefore, we estimate one global aspect ratio (w^*, h^*) from R for each iteration.

We use an intuitive two-stage process for determining the reference aspect ratio (w^*, h^*). First, we determine if the majority of instances has a horizontal, a squared, or a vertical aspect ratio. The reason for this first step is that similarly shaped aspect ratios may have different actual aspect ratios and we first want to determine the rough shape of the majority of instances before deciding which actual aspect ratio we should use. We then determine the one aspect ratio (w^*, h^*) of the majority of instances of the resulting aspect ratio type (horizontal, squared, or vertical). Note that for this procedure we exclude outlier instances which do not overlap more than 0.3 with the initial estimation. This happens if the initial rectangle is extremely small and the rectangle we choose on the grid, which has a minimum size of 100 HOG cells, hence cannot be expected to actually model the shape of the instance appropriately. For these instances the respective rectangles apparently could not be described by any of our aspect ratios (or on any of our scales).

We then only allow small deviation from the global aspect ratio (w^*, h^*). In our implementation, H_i only includes aspect ratios in

$$\begin{aligned} R' = \{&(w^*, h^*), (w^* + 1, h^*), (w^*, h^* + 1), \\ &(w^* + 1, h^* - 1), (w^* - 1, h^* + 1)\} \end{aligned} \tag{10.44}$$

Thus, the search space only includes aspect ratios which do not deviate more

than one HOG column or row from our global aspect ratio (w^*, h^*). We, however, do not allow "shrinking" with regards to area for the aforementioned reasons. Note that (w^*, h^*) and thus R' are updated during each iteration like the other latent variables.

Besides the aspect ratio, we also restrict the possible locations for our latent variables. We do not fully dismiss our initial estimations since this would lead to all instances i within the same image converging towards the same highest scoring rectangle. Therefore, we require that rectangle h_i must have a minimum overlap with the initial estimation $\hat{r}_i$ of 0.05. The same overlap is required with the rectangle defined by the latent variable of the respective previous iteration. In other words, our estimation may not converge completely away from either the initial estimation or the current estimation. Also, since none of these requirements directly considers the positive pixels found by the initial model, we also do not allow bounding boxes which have less than half the positive pixels than the initial bounding box. If no rectangle with an aspect ratio from R' is found for an instance which fulfills these requirements, we simply keep the previous rectangle for this instance.

Note that the smallest octave of our scale space described in section 7.1 yields the largest number of hypothesis compared to "lower" scales. In other words, on high image resolutions, the probability of detecting a false positive rectangle due to noise is higher. Therefore, we only use this octave in our search space if a large number (more than 25%) of rectangles found by our initial model yielded rectangles within this octave. In practice, otherwise dismissing the first octave prevents partial detections for datasets with small images and large objects (relative to the image size). This does not affect datasets with large images and relatively small instances.

For the hard negative mining in the cutting plane algorithm (see previous section), we now use the same limited set R' of aspect ratios since the most useful negative examples are intuitively the ones with similar aspect ratios as the desired object.

10.5.5 Post-Processing

We also perform two post-processing steps which improve the estimated rectangles h_i^* of the CCCP algorithm for some positive images. Mainly, images with very small object instances (relative to the image size) are affected by these steps.

Very small object instances may be too small to be detected with the

estimated aspect ratio on any scale. Usually such objects are detected on the scale which corresponds to the largest image resolution. For detections on this scale (and the one neighboring scale), we thus perform an additional search on a double-resolution version of the respective positive image by performing the inference of latent variables as explained above with the final model vector $\mathbf{w}$. As above, we require that we have a minimum overlap with the previous rectangle (i.e., the output of the CCCP algorithm) and the initial rectangle, whereas for this final step, we reduce the required overlap with the latter to 0.01. Also, we omit the requirement regarding the relative number of positive pixels. We can alleviate these requirements, since our starting point is already the output of the CCCP algorithm. For this search we keep the original result of the CCCP as hypothesis such that it is kept if no rectangle on the double resolution has a higher SVM score. The rectangle we find may, however, still be too large for very small instances (we still have a minimum size of roughly 100 HOG cells). Thus, for significantly boosting the accuracy of bounding boxes on such small instances, we shrink bounding boxes on the smallest scale by shifting the rectangle borders inwards until we meet a positive pixel as defined by the initial model. Note that this often results in boxes which are relatively similar to the initial model's boxes. In other words, for very small instances, we preferably trust the initial model.

Also, in a similar but scale-independent step, if the new rectangle h_i^* does not deviate from the initial rectangle $\hat{r}_i$ by more than one HOG cell on each side, we assume that the initial rectangle is a better estimation since the initial model works on pixel level. Therefore, in this case we snap h_i^* back to $\hat{r}_i$ as the final estimation for the object.

Finally, we apply non-maximum suppression, i.e., for pairs of rectangles with overlap > 0.85, we remove the one with the lower SVM score under the final model vector, unless both instances are on different scale octaves, i.e., at least 4 scales away from each other. Note that this step requires a scoring function which is not available for the initial model. The non-maximum suppression is an important step since it removes false positive detections of the initial model, especially in the relatively frequent case of two initial rectangles on the same object instance caused by disconnected positive pixel areas. Also, completely false positive detections of the initial model may converge to nearby true positive detections.

10.6 Evaluation

In this section we show the effect of using the latent structural SVM training algorithm on our overlap-recall curves. Note that the CCCP algorithm is HOG-based, thus we cannot expect much improvement on classes for which the initial color model is stronger than the initial HOG model, namely most classes from Oxford 17 Flowers. Also note that we now use the negative set form FlickrLogos-32 as negative training set for the Flower classes instead of using all other flower classes as for the initial model. The reason is that we perform hard negative mining on our negative set as explained above, so using other flowers as a negative set is not reasonable, since we expect the BOW histograms to be too similar among the negative set. Besides, the negative set of FlickrLogos-32 provides $6,000$ images and is thus significantly larger.

10.6.1 Brand Logos

In figure 10.3, the overlap-recall curves on FlickrLogos-6 are shown for the rectangles found by the CCCP algorithm and for the initial model for comparison.

We also show results for a different method (labeled "Objectness") which is discussed below. For our six selected classes, the CCCP algorithm yields the best results except for "Shell".

For "Aldi", the CCCP algorithm does not improve the results noticeably. For the "Shell" class the results of the CCCP algorithm are clearly inferior to the initial model mainly because the global aspect ratio estimated for "Shell" (which is also bound to HOG cells) is not optimal as shown in the example result in figure 10.6. At the same time for this class the initial model works exceptionally well, while the SVM model tends to produce partial detections.

In addition to the six classes used throughout the previous chapters, we also test our approach on the remaining 26 classes of FlickrLogos-32 for our final method. The results are combined into one overlap-recall-curve in figure 10.4. Example results are shown in figure 10.5 and 10.6.

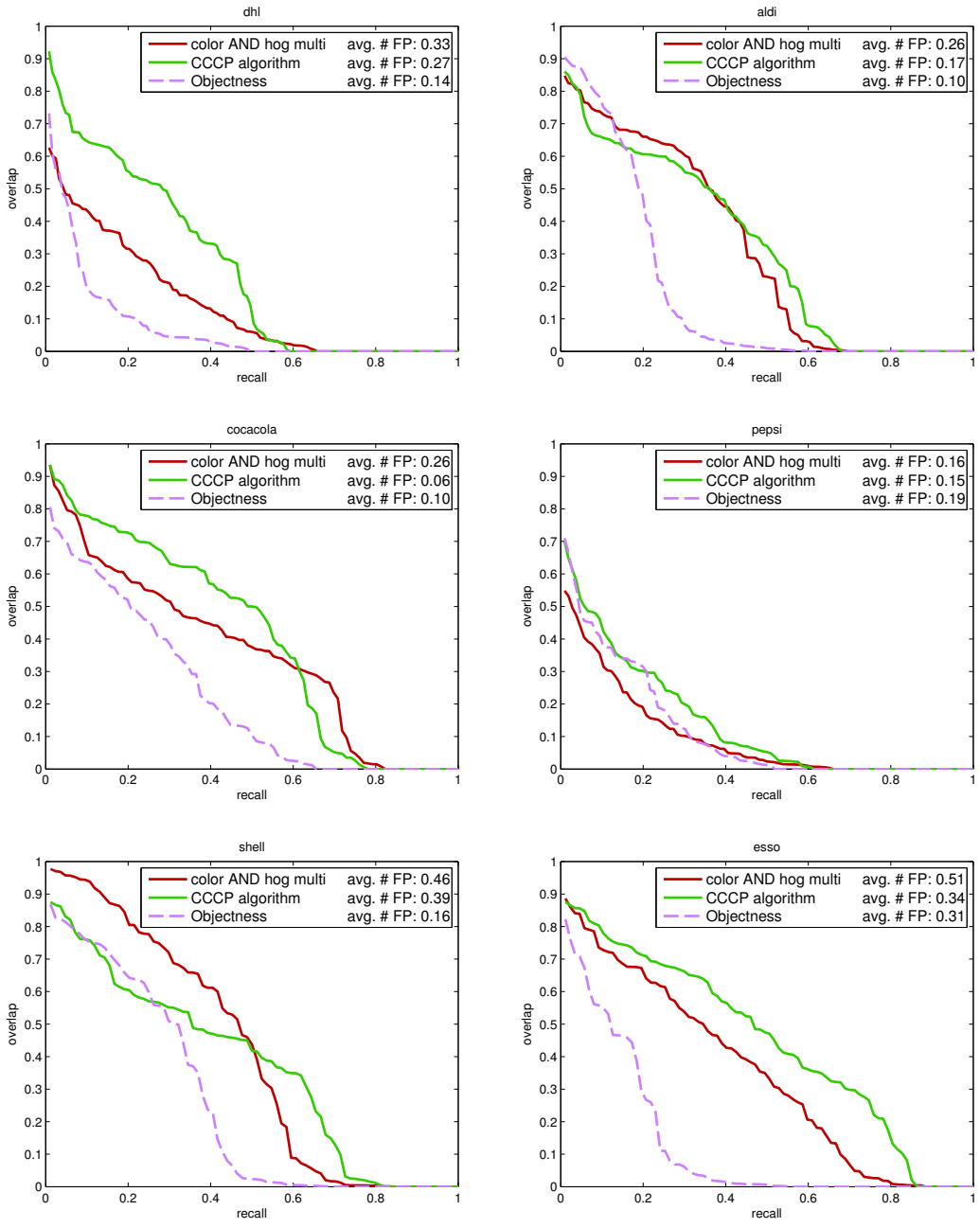


Fig. 10.3: Overlap-recall-curves for the six logo classes of FlickrLogos-6. Results of CCCP algorithm (green) and the initial model (red) are shown. The dashed purple curve shows results of applying the Objectness method.

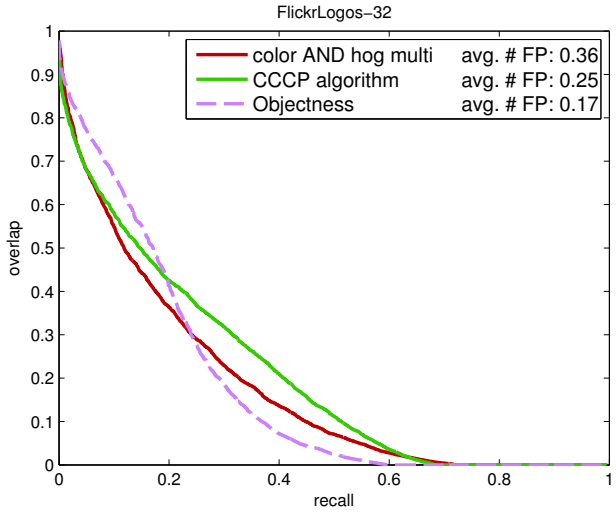


Fig. 10.4: OR curves over all 32 classes from FlickrLogos-32.

The examples in figure 10.5 are images where the CCCP algorithm yields better results than the initial model while for the examples shown in figure 10.6, the initial model yields better boxes. In each example, the red rectangles are the boxes found by the initial model. The green rectangles are returned by the CCCP algorithm.

Note that the lower right example of figure 10.5 shows a situation where a false positive detection by the initial model is removed during non-maximum suppression, since both initializations converge towards the same location. The left example of the center row shows a result where we obtain good boxes for very small instances due to our post processing.

On the full FLickrLogos-32 dataset, the CCCP algorithm also improves the results compared to the initial model with regards to the overlap recall curve in figure 10.4.

For some logos, the initial model yields better results, however. The reason for this is the tendency of the latent SVM to converge towards partial, but more distinctive objects or towards misleading background. The images in figure 10.6 are examples for this issue. In three of the shown examples, CCCP detects only the "most unique" portion of the respective logo, while the initial model also covers the surrounding part, which leads to a significantly superior overlap with the ground truth.

Fig. 10.5: Example results for CCCP algorithm on FlickrLogos-32 where CCCP improves the bounding boxes. Green rectangles indicate the CCCP results, while red rectangles are obtained from the initial model. In the lower right example, a false detection of the initial model is removed during non-maximum suppression.

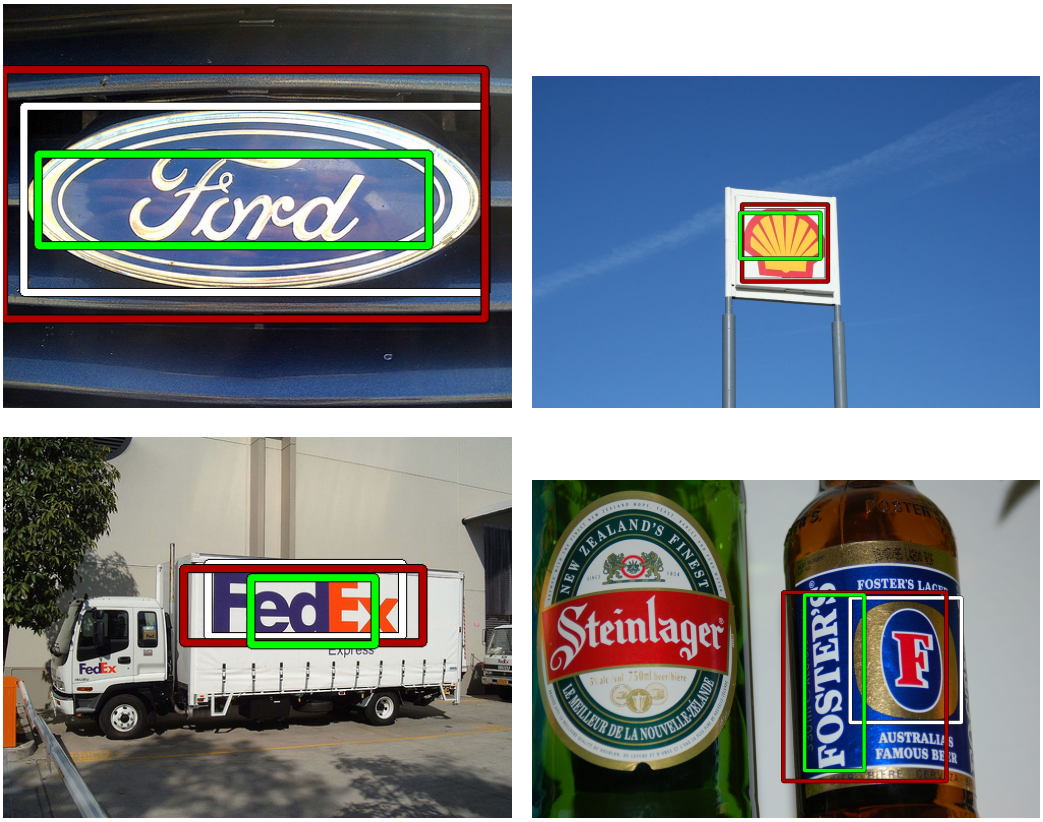


Fig. 10.6: Example results for CCCP algorithm on FLickrLogos-32 where the initial model is better than the CCCP results. Green rectangles indicate the CCCP results, while red rectangles are obtained from the initial model.

Note that in these cases the CCCP algorithm still achieves the goal of finding a distinct object which all positive images have in common. Depending on subjective ground truth interpretation, the detected part could also be considered the "correct" object. In the lower right example, a background region has a higher SVM score than the correct object. Also note that a better global aspect ratio would have improved the upper right result.

The CCCP algorithm also reduces the number of false positives compared to the initial model. The reason is that the CCCP algorithm may force multiple predictions of the same object from the initial model to converge towards the same rectangle which is then resolved by non-maximum suppression. This can be done since CCCP provides an SVM model, which

allows such non-maximum suppression based on score values as opposed to the initial model. The lower right example in figure 10.5 shows an example where an initial false detection is removed.

10.6.2 Flowers

As mentioned above, the Flowers dataset is not very suitable for gradient features. Thus, the CCCP algorithm which is based on HOG cells cannot improve the overlap-recall statistics for this dataset as shown in figure 10.7. The overall overlap even slightly decreases.

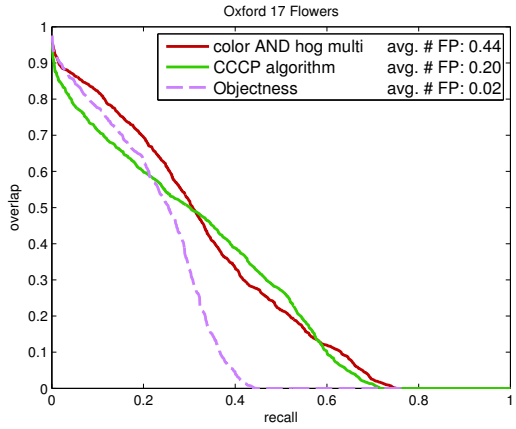


Fig. 10.7: Overlap-recall curves for initial model, Objectness method, and CCCP algorithm.

Again, a possible advantage of the CCCP algorithm is the lower number of false positive detections which means that multiple detections from the initial model converge towards a single object instance and the surplus detections are then removed during non-maximum suppression. It thus depends on the application of the bounding boxes whether applying the CCCP algorithm makes sense for this dataset.

Figure 10.8 shows a few example results. The examples reflect the fact that the HOG-based CCCP algorithm usually fails to find a bounding box which is better than the relatively accurate initial model. The reason is that the HOG features describing the outline of the flowers are relatively similar to the features in the background which violates one of our requirements stated in the beginning of this work. Note that in the lower right example, the

CCCP algorithm even removes one detection from the initial model which is actually a true positive detection.

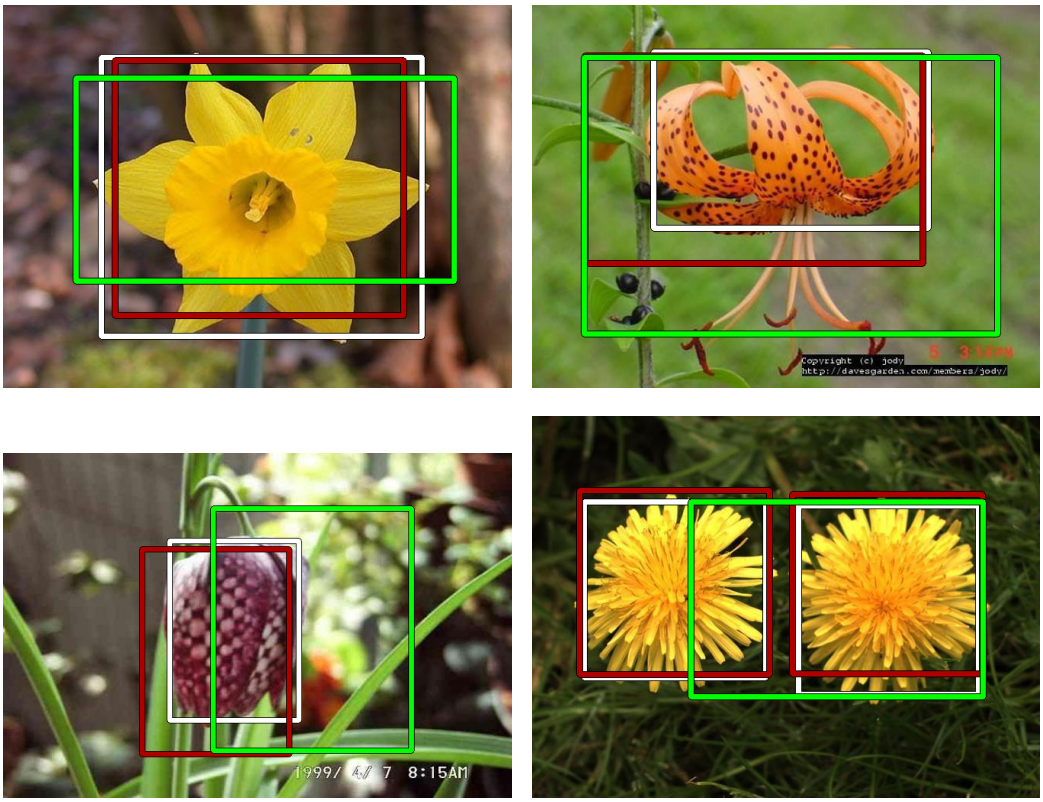


Fig. 10.8: Example results for CCCP algorithm on Oxford 17 Flowers. Green rectangles indicate the CCCP results, while red rectangles are obtained from the initial model.

10.6.3 3D Object Categories

Since the 3D Object Categories dataset consists of objects which provide relatively strong HOG features, it is more suitable for the CCCP algorithm. Still, as shown in figure 10.9 the results are not noticeably better than the initial model (with almost identical area under the curve), since the best overlapping bounding boxes (on the left of the plot) are slightly worse for the CCCP algorithm. In fact, however, for seven of the ten classes the CCCP algorithm outperforms the initial model to some degrees (with regards to area

under the curve), while on three classes the initial model is better. Therefore, we show additional plots in figure 10.10 where the classes are divided into the three classes where the CCCP algorithm does not improve the results (namely "Shoe", "Stapler", and "Mouse"), and the remaining seven classes where it does yield better bounding boxes.

A few qualitative example results are shown in figure 10.11. In the bottom images, again examples for detections of re-occurring characteristic sub-objects are shown as for both cars and bicycles the wheels are most characteristic.

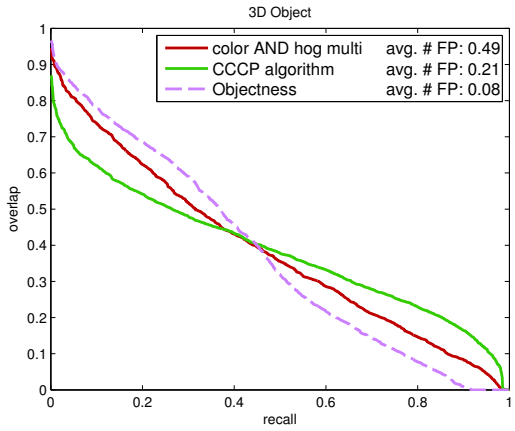


Fig. 10.9: Overlap-recall curves for initial model, Objectness method, and CCCP algorithm.

Comparison to Objectness Measure

In each of our overlap-recall-plots we also show a curve we obtain by using a method which is based on the "Objectness" measure introduced by Alexe et al. [1, 2]. The Objectness measure assigns a score to image windows indicating the probability that the window contains any object. The measure is based on five image features and uses a segmentation method by Felzenszwalb and Huttenlocher [19]. It requires training on manually annotated object bounding boxes which were taken from the 2007 VOC challenge dataset [16]. Experiments were conducted using a publicly available implementation.

It should be emphasized, that the Objectness measure is not designed for automatic annotation as defined in this thesis. The idea of the Objectness

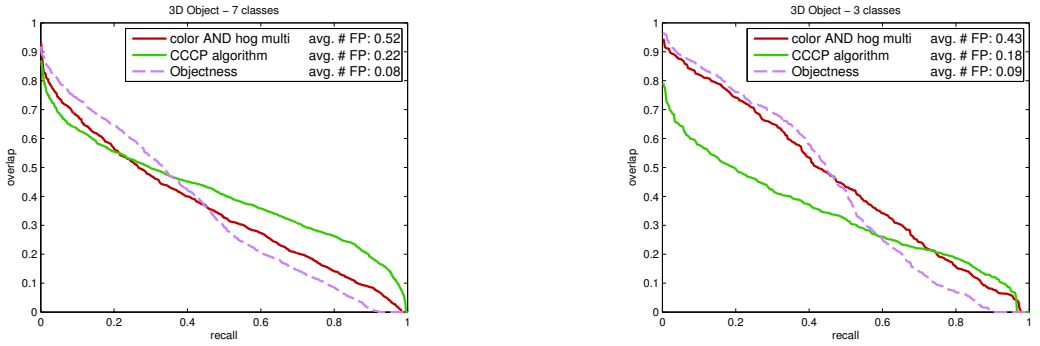


Fig. 10.10: Overlap-recall curves divided into seven classes where the CCCP algorithm outperforms the initial model and three classes where it fails to improve on the initial model.

measure is to detect any object in a given image while neither assuming nor exploiting the fact that we have a set of images which all show one common object. In other words, by applying the Objectness measure, one obtains salient image regions. This is the reason why we use it for comparison, since our approach does not aim at finding the most salient regions or the regions which are most likely to contain any object, but regions which consistently show the same object across all positive images. Therefore, it is reasonable to compare our approach to an approach which does the former.

In order to obtain comparable results, we simply determine the one rectangle within each positive image which is most likely to contain an object according to the Objectness measure. We then create an overlap-recall curve as for our approach. The resulting curves are shown in figures 10.3, 10.4, 10.7, and 10.9. For all curves, the results for the Objectness-based method are below the curves obtained from the CCCP results. Thus, we conclude that our approach goes beyond simply finding salient regions which happen to contain the desired object.

Note that for the 3D Object dataset, the objectness method is almost on par with the CCCP results with regards to area under the curve. The reason is that this dataset only consists of images showing one single object per image. Also, there is only some background noise present but usually no additional objects in the background. As a consequence, for this dataset automatic annotation as defined in this thesis and finding the most likely object location become almost equivalent tasks. The Objectness method

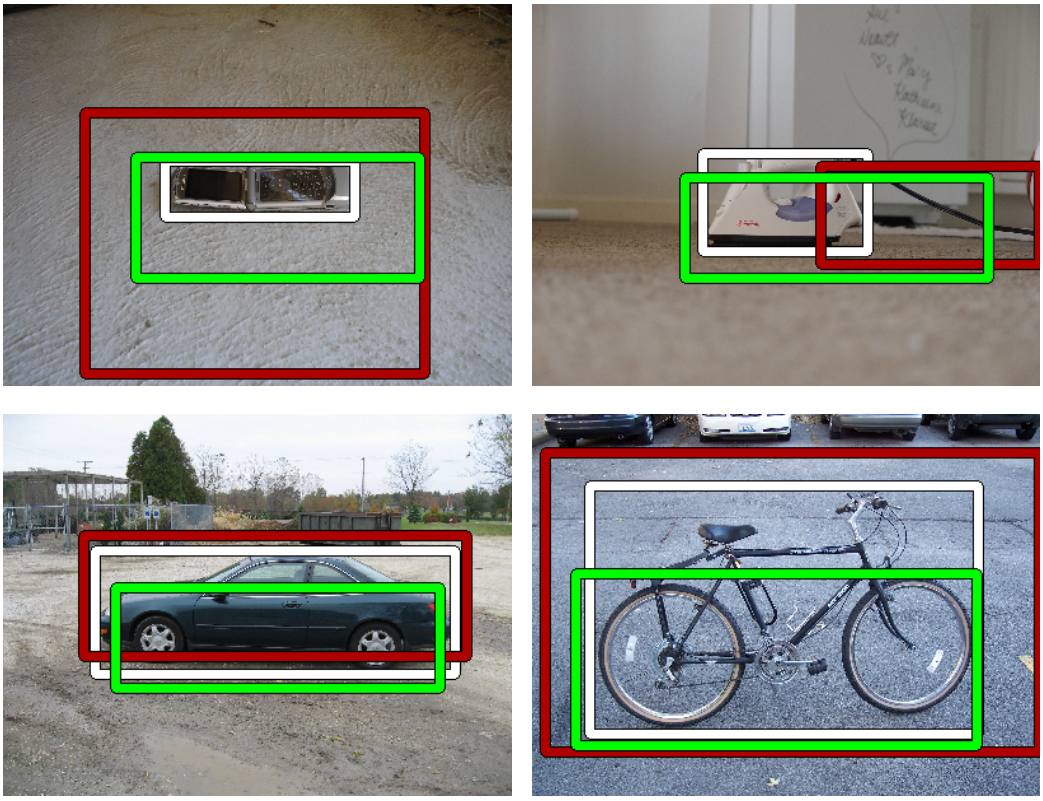


Fig. 10.11: Example results for CCCP algorithm on 3D Object Categories. Green rectangles indicate the CCCP results, while red rectangles are obtained from the initial model.

performs extremely well at the latter task, since it often detects exact object boundaries. On Flower classes with comparable properties (or where the initial model already fails), the Objectness method also clearly outperforms the CCCP algorithm.

It should be mentioned that in theory, we could accept multiple hypotheses (i.e., rectangles) from the Objectness measure, i.e., use the top k rectangles as automatic annotations. Increasing k quickly improves the overlap-recall-curves, since the desired object is usually among the top scoring image windows. At two hypotheses, the Objectness method is roughly on par with the CCCP algorithm for all datasets and outperforms it on 3D Object Categories. However, this also leads to a high number of false positive detections

which is almost equivalent to the number of true positive detections. Given the problem of automatic annotation, this is not a reasonable result. If we further increase k, the false detections considerably outweigh the true positive detections. For comparability with our approach, we thus only use the one top scoring hypothesis per image.

10.7 Discussion

In this section we discuss why the latent structured training algorithm is suitable for our problem. In its original form, the CCCP algorithm is designed for training an SVM model in the face of unknown properties of the training data. In [53], the CCCP algorithm is shown to be effective for training an SVM model for object detection with latent variables.

Our problem of automatic annotation is obviously related to the problem of training an SVM model for object detection, even though our setup is different from [53]. The fact that our goal is not to train a classifier, but to find the best configuration of latent variables only constitutes a marginal difference, since the algorithm performs both. The bounding box estimations are updated during each iteration of the algorithm and finding the optimal configuration for the latent variables is explicitly part of the optimization problem stated in equation 10.21.

In other words, the algorithm searches for better rectangles in the positive examples and - based on these rectangles - re-trains the prediction model which is then used for new predictions. The CCCP algorithm can thus be seen as an instance of Expectation Maximization algorithm which basically estimates unobservable model parameters by iteratively building a model based on current estimations and then re-estimating said parameters based on this model.

Intuitively speaking, by building a model vector, the CCCP algorithm iteratively learns a consistent description of all positive examples. The initial statistical model, in contrast, does not consider any consistency between the estimated rectangles but only seeks consistent individual discriminative features. Afterwards, each image is treated individually by the initial bounding box estimation. As a consequence, the initial model often includes background areas, whereas the CCCP algorithm can base its estimation for each image on the estimations from all other positive images and thus better determine areas which do not belong to the actual object. The examples in fig-

ure 10.5 illustrate this property since the bounding box found by the CCCP algorithm usually removes surplus background areas around the desired objects. Besides, the CCCP algorithm lowers the number of false detections as discussed above.

Another important property of our training algorithm is its built-in bootstrapping mechanism. As explained above, the cutting plane algorithm used for optimization implicitly performs hard negative example mining by finding the most violated constraints during each iteration which is an advantage of the structural SVM formulation. Thus the algorithm allows us to consider a very large set of negative examples, i.e., all possible rectangles of all negative images and selects the most useful subset (at most one per image) which contribute constraints for the optimization problem.

It should also be emphasized that in a purely structured problem, the loss function for object detection may be defined based on the degree of incorrectness of a prediction. This strategy is used by Blaschko et al. [7]. The loss function on positive examples then depends on how much a predicted rectangle overlaps with the actual rectangle (which is considered correct and not latent in their situation). This allows creating constraints for rectangles within positive images that are only "partially" positive, which is in turn reflected by the loss function on the right side of the respective constraint (see equation 10.19). Even though this is a promising strategy, we cannot adapt it for our problem for two reasons. First, it would require the loss function to depend on the current estimation for latent variable h_i, since in our case h_i defines the current rectangle of positive example which is needed for computing the overlap for such a loss function. Therefore, formally requirement 10.27 would be violated. Second, this loss function would feed our algorithm inaccurate information beyond the potentially inaccurate initial bounding boxes since it would weight different false positive detections in positive images differently even though they may all have zero overlap with the actual object. For illustration, consider a rectangle h_i which covers the desired object and a large background region. For this rectangle, obviously many predictions exist which do not overlap with the actual desired object at all, but may still have a relatively large overlap with h_i and thus obtain a "good" loss value.

These observations also influenced our decision to use a bag-of-words histogram which only counts the appearance of visual features and ignores their exact location. Thus, inaccurate detections, i.e., bounding boxes which are too large or small but still contain or lie on the desired object, still

yield some correct components to the feature vector and hence to the trained model. Recall that we often obtain over-detections from our initial model.

11. EVALUATION OF APPLICATIONS FOR AUTOMATIC ANNOTATIONS

In the previous evaluation, we have only considered how much overlap our annotations achieve with manual bounding box annotations. This is an obvious evaluation scheme for the quality of bounding boxes. However, this is just a technical performance, which might say little about the usefulness in an actual application. Therefore, we use our automatically determined annotations for two common applications: Image retrieval and image classification.

11.1 Evaluation for Image Retrieval

In this section we evaluate our automatic annotations in an image retrieval scenario. Image retrieval usually means the task of finding images showing a specific object or concept among a very large set of images. There are two main retrieval variants which differ in the type of query issued by the user. Either the desired object or concept is defined by a query term or by a given query image which contains the respective object (also known as query-by-example). For our evaluation we consider the latter variant.

More specifically, the task we examine is the task of finding images within a large database which show the same object as a given query image. Also, we only consider visual features and no meta-information such as image tags.

Usually, for query-by-example methods, a so-called inverted index is built for the database to be searched. An inverted index maps cluster ids of visual words to a list of images within the database which contain said visual word. This enables a quick search for all images in the database which contain a certain visual word.

It is therefore helpful if the database images can be reduced to annotated regions of interest (e.g. bounding boxes) which show the desired object before indexing. The idea is to ignore all image features which lie outside the

annotated regions, since these features are considered background features. As a consequence, positive images then occur less frequent in the inverted index for cluster ids of background features, because background features do no longer appear in the positive images.

The retrieval approach we use for evaluating our bounding boxes is the brand logo retrieval system by Romberg and Lienhart [35]. We hence use a bag-of-words model on RootSIFT features with tf-idf weighting and burstiness measure.

An inverted index is created for a database of images that holds both logo images as well as logo-free images whereas we use the data split into query and indexed images as suggested by the authors of FlickrLogos-32. We then perform query-by-example retrieval, i.e., we use the inverted index in order to find images containing the same visual words as a given query image. The retrieval result is hence a list of images sorted by the number of visual words they have in common with the respective query image.

One widely used evaluation method for retrieval methods is using the top-k retrieval results for a k-nn classification algorithm. Thus, we perform a majority voting among the top k retrieved images which are the first k images from the aforementioned list of retrieved images. In other words, the top k images are the k images from the database which have the most visual words in common with the query image. If the majority of these top k images show the same logo as the query images we count the respective query image as a true positive detection. The relative number of true positive query images then yields a recall value.

Specifically we compare four scenarios which differ in the way how our database, i.e., our index, is created. First, we extract features from the full images. Second, we only extract features from the regions determined by our automatic object annotation method. Third, features are only taken from regions obtained by manual labeling of human experts at pixel-level. For comparison, we also use the Objectness-based boxes.

Note that, strictly speaking, in a realistic scenario, we could not use the respective query images for the automatic annotation process. For simplicity, however, we do not remove each query image before retrieval but simply use the boxes we obtain from the previous experiments, based on all images including the respective query image. Of course, the query images are still not included in the retrieval database. Since the automatic annotation process is fully independent of the retrieval (which also uses different visual features), and the approaches we use for comparison, namely full images, ground truth

k	1	3	5	7	9
full images	0.83	0.81	0.79	0.78	0.77
Objectness	0.84	0.81	0.80	0.78	0.78
our boxes	0.86	0.83	0.82	0.81	0.80
human annotations	0.87	0.85	0.85	0.84	0.84

Tab. 11.1: Average recall of k-nn classification on retrieval results for FlickrLogos-32.

boxes, and Objectness, are not based on a training set for devising bounding boxes, it is justifiable for our experiment not to remove each query image for the annotation process.

Table 11.1 states the average recall of the k-nn classifier over all logo classes for running our k-nn classifier over the top-k retrieval results for each of the 960 query images. Note that the database also contains non-logo images which may also affect the recall of a k-nn classifier, since they may be detected with a higher similarity than images actually showing the desired object of the query image. Our bounding boxes improve the performance compared to retrieval on full images and is inferior to manual annotations.

This result shows that our automatic annotations are capable of removing distracting background regions from the images in our database. It is not surprising that the manual pixel-wise annotations yield the best results, because they remove everything from the positive images except the actual true positive features.

11.2 Evaluation for Classification

In this section we use our automatic annotations for an image classification scenario. The task is to decide whether an image region belongs to an object class. For this experiment, we again use the FlickrLogos-32 dataset.

The classification task first requires training a classifier. We simply use a linear SVM based on the SVMLight [21] implementation.

As our negative training set, we use $1,000$ images from the negative (non-logo) set of FlickrLogos-32. Since the negative set is considered to consist of "random" images, we extract bag of word histograms from the full negative images as our negative training vectors.

For the positive training set we again compare four variants: First, we simply use the full positive images from the training and validation set {train-val} of the respective logo class (i.e., 40 images). Thus, for this variant we assume we do not have any annotations. Second, we extract our training examples based on manually annotated bounding boxes which is the optimal training set for this scenario. Third, we run our automatic annotation approach on {train-val} (whereas we exclude the test set in contrast to our previous experiments) and use the resulting bounding boxes for extracting positive training vectors. Finally, we use the boxes obtained from the Objectness measure as training examples.

Our negative test set consists of $10,000$ images from World 100k (ignoring extremely small images leaves us with $8,341$ test images). As positive test set we use all instances from the test set of the respective logo class. Since we want to determine the classification performance, the instances are extracted based on manual annotations, i.e., our positive test set is guaranteed to consist of actual true positive image regions.

For all test and training instances we extract the BOW histograms as described in section 10.5.1 and 10.5.4, i.e., we search the one rectangle among all image scales which has a reasonable aspect ratio and the highest overlap with the manually annotated bounding box. Therefore we only have histograms in our test and training sets based on a similar number of HOG cells and thus prevent any bias towards small (or large) boxes. As a consequence, we ignore test and training examples where the object instance is too small to be modeled by one of our aspect ratios with more than 0.5 overlap, since otherwise the modeled region cannot be considered an actual positive example.

Note that using the manual bounding boxes as test set is arguably not a realistic situation, since these annotations cannot be assumed to be available for the test set. However, for the sake of testing the classification performance, it is reasonable to use the actual true positive test examples.

For evaluation, we again use ROC curves. For creating a ROC curve, we consider the SVM distance from hyperplane as classification score for each test instance and then sweep a threshold over these values, counting the ratios of true positive and false positive classifications for each threshold value. Note that we cannot draw one meaningful curve over multiple classes like for instance for the color model, since for the SVM classifier, the score values, i.e., distances from hyperplane, are not comparable across multiple classes. In figure 11.1 we show a few ROC curves for selected classes from

FlickrLogos-32. In each plot, the green curve is the result of the SVM trained on automatic annotations. The dashed black curve is the result of the SVM trained on ground truth annotations and the blue curve is obtained when using full images as positive training set.

Note that some classes only yield very small positive test sets, since only a few instances are present which are large enough to be modeled as explained above. For some classes we hence only obtain curves with limited expressiveness, since the resulting curves are almost rectangular for all three variants. We therefore show a selection of classes where the difference between the four variants is more pronounced.

For each curve shown in figure 11.1 the SVM trained on our automatically annotated boxes yields a ROC curve which is closer to the baseline (SVM trained on manual annotations) than the results of the SVM trained on full images. The only exception among these example classes is the class "Starbucks" (the lower right plot) where our automatic annotation does not yield positive training examples which are better than the full images. Yet, training on full images is only better than training on the automatic boxes for 4 of the 32 classes. Also for "Starbucks", the Objectness-based annotations yield an SVM with superior classification results with regards to area under the ROC curve in comparison to the automatic boxes. This is the case for 13 logo classes, while for 18 cases the automatic boxes yield a better area under the curve. For the remaining classes, both results are identical (or no test examples could be modeled by our templates). Note that, as mentioned above, the differences between the Objectness method and the automatic boxes are only very small for many classes due to the relative low number of test examples. Overall, the usefulness of additional positive training instances is hence limited, i.e., in this experimental setup a few relatively good positive examples (as provided by the Objectness method) are often sufficient.

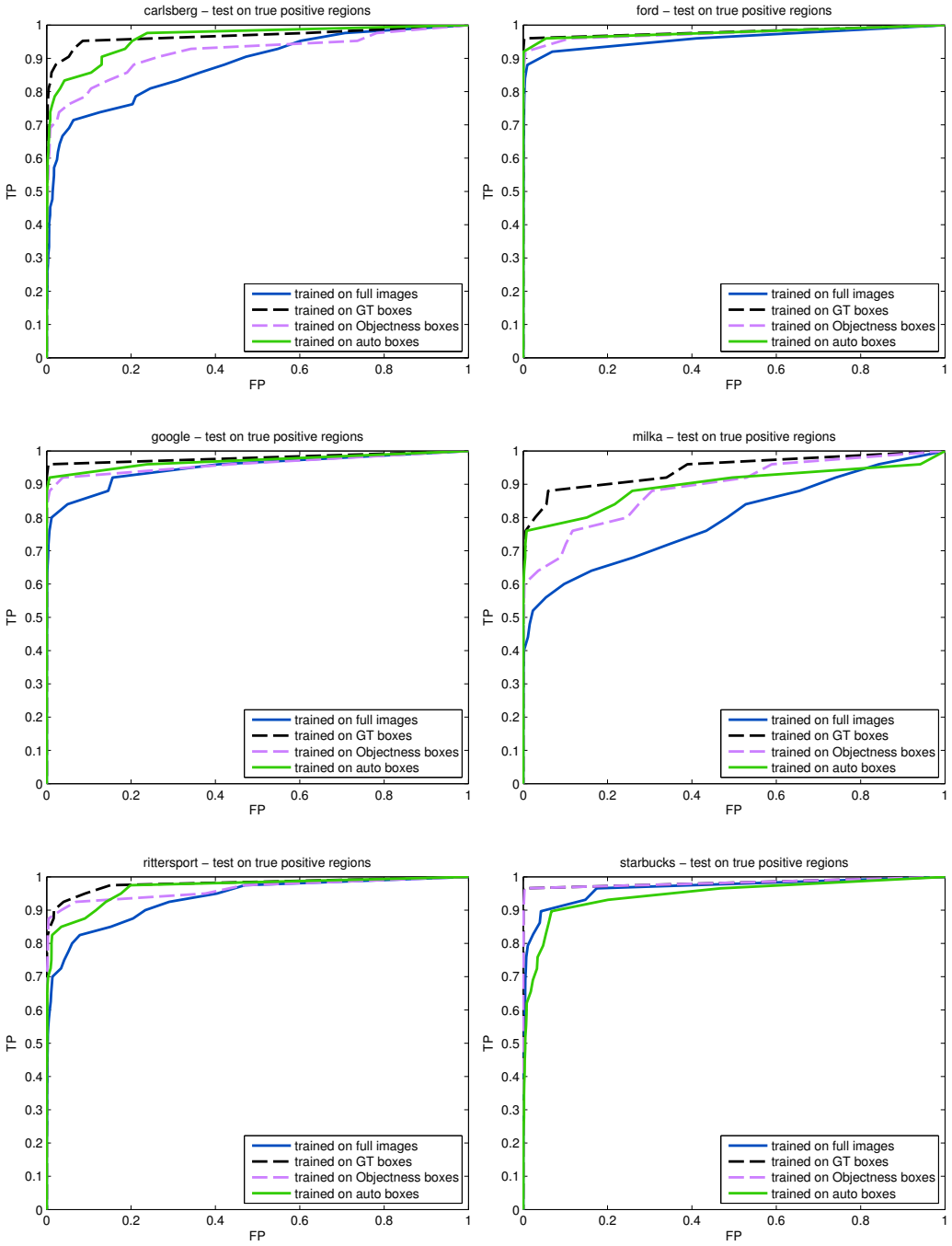


Fig. 11.1: ROC curves for classification experiment.

12. SUMMARY AND CONCLUSION

In this chapter we summarize the thesis and end it by discussing the main aspects and issues of the approach and by giving some outlook on possible future work.

12.1 Summary

In this thesis, we have introduced and discussed a method for automatic object annotation in weakly labeled images. We have assumed that sets of positive and negative images were given and that all positive images should show at least one instance of a common object class.

For our initial approach we have proposed a statistical model based on feature statistics which is solely derived from statistics over global image labels. The basic idea of the initial model is examining our confidence in a given feature being not discriminative for an object class based on our observation on the negative set. We have implemented this initial model for two feature types, namely colors and visual words based on HOG features. Also we suggest a heuristic for estimating bounding boxes based on discriminative features.

Afterwards we have discussed a latent structured learning algorithm for improving our bounding box estimations. We use an instance of the CCCP algorithm which simultaneously trains an SVM model based on Bag-of-Visual-Words histograms and estimates bounding box locations in the positive images. The idea is thus to build a model vector which describes the positive objects by means of a occurrence histogram over visual words (i.e., local visual features) which then can be used for searching for better bounding boxes. Since the model vector is trained on all positive instances in each iteration of the CCCP algorithm, we expect it to converge towards a weight vector which rewards the co-occurrence of multiple features which are indicative for the wanted object while at the same time learns a penalty for local

features which are not common across all images and therefore belong to the background.

In our experiments we have shown that for a number of test classes from different datasets, we find promising bounding boxes albeit leaving room for improvement. We have also demonstrated that the automatic boxes are useful for different applications.

12.2 Outlook

In this section, we briefly address and discuss some of the main issues and possible starting points for future work.

According to our definition of the problem of automatic object annotation, we assume no knowledge beyond global image labels. This is the reason why our approach overall involves a number of heuristics which are relatively straightforward methods.

This applies for instance to the feature merging process which intersects positive pixels of multiple features without any weighting regarding the usefulness of the respective feature. Especially if we were to add further features to the process, the intersection operator, which equally considers the "votes" of each feature for each pixel, is likely to fail.

Another component which could be replaced by a more sophisticated method is the heuristic which estimates multiple bounding boxes from positive pixels which also does not consider any information beyond the two-dimensional distribution of positive pixels. Note that an overall stronger initial model may allow different and less complex methods (in comparison to the relatively expensive CCCP algorithm) for improving the estimations.

Also, the approach presented in this work requires a considerable number of parameters which have been selected empirically, for instance the definition of reasonable aspect ratios for rectangles or thresholds for various post-processing step of our bounding boxes. If we assume we have no validation set with manual ground truth bounding boxes, we cannot determine optimal values for these parameters for different classes. One especially important example is the factor α used for our dynamic threshold for the initial model. The idea behind this factor is to formulate a model which is independent of the actual object class and feature type. As a consequence, this means that it is assumed to work for any arbitrary object class which is not realistic and hence also not true for all classes of our datasets. Experiments show,

however, that selecting parameter values specifically for each given class may lead to significantly better results of our approach. Therefore, this is an obvious starting point for enhancements.

We also want to emphasize that the assumption that features which are not very suitable for a dataset yield a weak reference feature and thus a large set of positive features which in turn leads to over detection of the desired object does not always hold. As already discussed in section 8.1, if there is one especially strong feature, which is not sufficient for detecting the respective object, the merging approach may fail. For instance, a few object classes from the 3D Objects Categories dataset have strong color features which, however, only describe very small areas of the respective objects. While satisfying our requirements for discriminative features, these colors are not sufficient for devising good bounding boxes. Similarly, some flower classes have relatively strong HOG features which actually lie on the flower instances but are relatively small and thus remove true positive image regions. However, these cases are related to cases where we detect actual discriminative sub regions of object instances which does arguably not contradict our problem definition, since object instances are not defined as the largest possible image regions all positive images have in common. It is therefore difficult to clearly distinguish these cases but worth examining.

12.3 Conclusion

It should, in conclusion, be emphasized that the approach presented in this work does not ultimately solve the problem of automatic object annotation. As discussed above, the results leave room for improvement with regards to overlap with manual annotation. However, it is also arguable if manual bounding boxes are always the best references for evaluation as also already discussed.

Besides, it should be mentioned that the object classes we use for evaluation are relatively suitable for our approach since they satisfy our assumptions mentioned in the introduction to certain degrees. In other words, the methods described in this thesis cannot be expected to work on arbitrary object classes and arbitrary sets of training images. Still, our results show that for a considerable number of different objects our approach yields reasonable results and also some aspects of the approach are apparently adaptable to other scenarios.

With regards to the aforementioned difficulties and considering the small amount of previous knowledge we require, we conclude that the results of our method are promising. As our overlap-recall curves indicate, we usually at least partially detect object instances or simply over detect object instances, especially if we only use the initial model. Both results can be useful in many scenarios. We also stress that our approach removes large amounts of background from many positive images which cannot be concluded directly from the overlap-recall curves, since they ignore true negative areas.

Also, in our experiments on retrieval and classification we show that the automatic annotations often improve the respective results compared to not using any annotations and are in some cases even close to manual annotations. Since the automatic annotations do not impose any additional manual work, we hence consider them a reasonable option for such tasks. Overall, automatic object annotations are undoubtedly useful. Therefore, the task of automatic object annotation is a highly relevant problem with a broad field of possible approaches and research directions for which this work may serve as a starting point.

APPENDIX

A. SVM OPTIMIZATION PROBLEM

In this section we give an intuitive, geometric explanation of the derivation of the SVM optimization problem. This explanation is based on explanations found in [6] and [39] and aims at intuitively deriving the dual problem for our special case of optimization. In [39], a mathematical derivation can be found. The optimization problem we want to solve is finding a vector $\mathbf{w}$ and bias b which minimize the margin of a decision function $f(x) = \langle \mathbf{w}, x \rangle + b$ under n constraints:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \tag{A.1}$$

$$s.t. \;\; \forall i \in \{1, ..., n\} : y_i(\langle \mathbf{w}, x_i \rangle + b) \geq 1 \tag{A.2}$$

Note that for clarity we subsume positive and negative training examples to a total of n examples as opposed to section 10.1.1 wehre we have n positive and m negative examples. Otherwise, equations A.1 and A.2 restate the optimization problem of equations 10.4 and 10.5.

This formulation of the problem is commonly known as the *primal formulation* of the optimization problem where the variables $\mathbf{w}$ and b, which must be optimized, are the *primal variables*.

The constrained optimization problem is defined by an objective function in equation A.1 and inequality constraints in equation A.2. Our goal is now to explain why we can express the solution to this optimization problem by a weighted sum over all n training vectors (whereas only the support vectors will have weights $\alpha_i > 0$):

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{A.3}$$

This solution can be derived from the *dual problem* formulation which is based on the method of Lagrange Multipliers which is sketched in the following sections.

A.1 Lagrange Multiplier Method

For deriving the dual problem formulation for constrained optimization, we first give an intuitive geometric explanation of the method of Lagrange Multipliers. Note that for the following explanations we always assume that our optimization problem does have a solution, i.e., all constraints can be satisfied by some point in the respective space.

A.1.1 Single Equality Constraint

Before dealing with the actual optimization problem with n inequality constraints we consider the problem of an optimization problem with one equality constraint. Suppose we want to minimize a function $f(\mathbf{w})$ subject to an equality constraint $h(\mathbf{w}) = 0$, i.e., our optimization problem is

$$\min_{\mathbf{w}} f(\mathbf{w}) \tag{A.4}$$

$$s.t.\ \ h(\mathbf{w}) = 0 \tag{A.5}$$

Let K be the number of dimensions of $\mathbf{w}$.

Note that a requirement for the method of Lagrange multipliers is that functions $f(\mathbf{w})$ and $g(\mathbf{w})$ have continuous first partial derivatives which holds for our problem defined in equations A.1 and A.2. Also note that in general, we allow constraints of the form $h'(\mathbf{w}) = c$ which we can re-write as $h(\mathbf{w}) = h'(\mathbf{w}) - c = 0$, i.e., equality with a constant $c \neq 0$ is implicitly allowed.

In the case of one constraint, the optimal point in K-dimensional space must yield the minimum value of $f(\mathbf{w})$ among all $\mathbf{w}$ satisfying the constraint $h(\mathbf{w}) = 0$. For a single constraint, the optimal point is intuitively the point where $h(\mathbf{w})$ tangentially touches $f(\mathbf{w})$ since for every other point on $h(\mathbf{w})$ there is a better value of $f(\mathbf{w})$.

For illustration, consider a parabola $f(\mathbf{w})$ over a two-dimensional space and a linear constraint $h(\mathbf{w}) = 0$. Geometrically, the line defined by $h(\mathbf{w})$ intersects the parabola at different height levels whereas the lowest of these height levels contains the wanted point where $f(\mathbf{w})$ has minimum value and $h(\mathbf{w})$ is still satisfied. At a tangential point of two functions, the derivatives of both functions must be parallel, so in the optimum point

$$-\nabla f(\mathbf{w}) = \beta \nabla h(\mathbf{w}). \tag{A.6}$$

for some value of β. Solving for 0 yields the derivative of the *Lagrangian function* $L(\mathbf{w}, \alpha)$:

$$\nabla L(\mathbf{w}, \alpha) = \nabla \left(f(\mathbf{w}) + \beta h(\mathbf{w}) \right) = 0 \tag{A.7}$$

where β is the *Lagrangian multiplier*. Thus, we need to find the point in which the Lagrangian function is 0 in order to find the constrained optimum of $f(\mathbf{w})$.

Note that in general, the Lagrange condition formulated in equation A.7 is only a necessary condition for optimum points. If we for instance consider a parabolic constraint over two dimensions and a (non-constant) two-dimensional linear objective function (i.e., a surface), we will find two points which satisfy the Lagrange condition, i.e., we have two points where the gradients of both functions are parallel.

A.1.2 Multiple Equality Constraints

Now we consider the case of a optimization problem with multiple equality constraints.

$$\min_{\mathbf{w}} f(\mathbf{w}) \tag{A.8}$$

$$s.t. \forall i \in \{1, ..., n\} : h_i(\mathbf{w}) = 0 \tag{A.9}$$

Again, we give a geometrical explanation for the Lagrange method. Intuitively, all potential optimal points must have the same value of $f(\mathbf{w})$ (otherwise there would be another optimum). In other words, we must demand that the gradient of $f(\mathbf{w})$ in our wanted point is orthogonal to the directions in which we are allowed to "move" on $f(\mathbf{w})$ without changing its value.

For example, on a parabola over a two-dimensional space, we know that the optimum must be fixed to one circular contour line of the parabola, otherwise we would allow moving "further down", i.e., along $-\nabla f(w)$ to a better value. Only points are allowed which lie on the circular contour line and "moving" along the circle does not change the value of the function.

At the same time, when searching the optimum point, we may only consider points satisfy the constraints, i.e., points from which we are only allowed to "move" in directions where the constraint does not change its value. Again, this direction is perpendicular to the gradient of the constraint. In

other words, the gradients of the constraints span a space in which every point is an invalid point.

Overall, in an optimal point for the objective function which satisfies all constraints (a *constrained optimal point*), the direction in which the value of the objective function may improve (i.e., the direction of $-\nabla f(\mathbf{w})$), must be forbidden by the constraints. Thus, in an optimal point, the direction $-\nabla f(\mathbf{w})$ must lie in the space spanned by the gradients.

For illustration, consider the aforementioned example of a parabolic objective function $f(\mathbf{w})$ over a two dimensional space where $-\nabla f(\mathbf{w})$ is the negative gradient pointing towards the minimum of $f(\mathbf{w})$. Let $h_1(\mathbf{w})$ and $h_2(\mathbf{w})$ be two linear constraints (and suppose for the sake of illustration that we ignore that the solution is trivial in two-dimensional space for two linear constraints). In the constrained optimal point, i.e., the point minimizing $f(\mathbf{w})$ while satisfying $h_1(\mathbf{w})$ and $h_2(\mathbf{w})$, following $-\nabla f(\mathbf{w})$ towards the minimum must be prevented by $h_1(\mathbf{w})$ and $h_2(\mathbf{w})$. In other words, in a constrained optimal point, following $-\nabla f(\mathbf{w})$ must cause violation of $h_1(\mathbf{w})$ or $h_2(\mathbf{w})$ which means that at a constrained optimal point, following $-\nabla f(\mathbf{w})$ for an infinitesimally small distance must lead to a point which lies in the space spanned by the gradients of the constraints, since this space contains all points violating any of the constraints.

Points $\mathbf{w}$ in the space spanned by the gradients of $h_1(\mathbf{w})$ and $h_2(\mathbf{w})$ can be expressed by a linear combination of $\nabla h_1(\mathbf{w})$ and $\nabla h_2(\mathbf{w})$ with coefficients β_1 and β_2. If we want to express that $-\nabla f(\mathbf{w})$ must lie in this space, we can write

$$-\nabla f(\mathbf{w}) = \nabla \beta_1 h_1(\mathbf{w}) + \nabla \beta_2 h_2(\mathbf{w}) \tag{A.10}$$

which leads to the Lagrange condition for two equality constraints:

$$\nabla L(\mathbf{w}, \beta_1, \beta_2) = \nabla f(\mathbf{w}) + \nabla \beta_1 h_1(\mathbf{w}) + \nabla \beta_2 h_2(\mathbf{w}) = 0 \tag{A.11}$$

Adding further constraints to these equations then yields the Lagrange condition for constrained optimal points for objective function $f(\mathbf{w})$ subject to n constraints (as defined in equations A.8 and A.9):

$$\nabla L(\mathbf{w}, \beta_1, ..., \beta_n) = \nabla f(\mathbf{w}) + \sum_{i=1}^{n} \nabla \beta_i h_i(\mathbf{w}) = 0 \tag{A.12}$$

A.1.3 Inequality Constraints

Now we replace the equality constraints by inequality constraints. Our new optimization problem becomes

$$\min_{\mathbf{w}} f(\mathbf{w}) \tag{A.13}$$

$$s.t. \forall i \in \{1, ..., n\} : g_i(\mathbf{w}) \leq 0 \tag{A.14}$$

The SVM optimization problem in equations A.1 and A.2 is such a problem. Note that the constraints of equation A.2 can be re-written to the form in equation A.14. As is common in literature, we denote Lagrange multipliers by α_i for inequality constraints.

Changing equality constraints to inequality constraints introduces one major difference. Intuitively, inequality constraints are less strict since they define feasible regions in which our optimum is allowed while equality constraints require that the optimum lies on the constraint function.

If the actual optimum of the objective function satisfies a constraint, the presence of this constraint obviously does not change the solution of the optimization problem. In this case, the corresponding Lagrange multiplier is 0 and the constraint is said to be *inactive*.

In the context of equality constraints, this means that if all constraints are inactive (i.e., if the optimum of $f(\mathbf{w})$ satisfies all constraints), and thus $\beta_i = 0$ for all i, the Lagrange condition of equation A.15 is reduced to stating that the optimum must be the optimum of the objective function.

Note that inequality constraints become inactive if they define a feasible region which contains the optimum of the objective function. Therefore, we can state that an inequality constraint can either be inactive or the respective feasible region does not contain the optimum of the objective function. In the latter case we can now postulate that the constrained optimum must be at a point where the negative gradient of the objective function $-\nabla f(\mathbf{w})$ "points away" from the feasible region. Otherwise, if $-\nabla f(\mathbf{w})$ points into the feasible region, we know that there is a better point which is closer to the optimum of the objective function.

For illustrating this issue, consider an inequality constraint which describes a parabola over a two dimensional space. Let the objective function also be a parabola. The constraint then defines a circular feasible region. If the optimum of the feasible function is inside this circular region, the

constraint becomes inactive, since it is already satisfied by the actual optimum of the objective function. Otherwise, there will be two points where the Lagrange condition holds, whereas in one point we will have a Lagrange multiplier $\alpha_{i,1} < 0$ and in the other one $a_{i,2} > 0$. The multiplier $\alpha_{i,1} < 0$ indicates a point where $-\nabla f(\mathbf{w})$ points into the feasible region while at the point where $\alpha_{i,2} > 0$, $-\nabla f(\mathbf{w})$ will point away from the feasible region (according to equation A.6). Intuitively, the latter point is closer to the actual minimum of the objective function. Therefore, we should chose the point in which $\alpha_{i,2} > 0$. Note that this is not the case for a circular equality constraint since if the optimum of the objective function is inside the circle, there will be no solution for $\alpha > 0$.

Overall, for inequality constraints, we are hence searching for points satisfying the Lagrange condition with a Lagrange Multiplier $\alpha > 0$ if they are active or $\alpha = 0$ otherwise. This leads to the following conditions:

$$\nabla L(\mathbf{w}, \alpha_1, ..., \alpha_n) = \nabla f(\mathbf{w}) + \sum_{i=1}^{n} \nabla \alpha_i g_i(\mathbf{w}) = 0 \tag{A.15}$$

$$s.t. \forall i \in \{1, ..., n\} : \alpha_i \geq 0 \tag{A.16}$$

$$s.t. \forall i \in \{1, ..., n\} : g_i(\mathbf{w}) \leq 0 \tag{A.17}$$

$$s.t. \forall i \in \{1, ..., n\} : \alpha_i g_i(\mathbf{w}) = 0 \tag{A.18}$$

These conditions are known as the Karush-Kuhn-Tucker (KKT) conditions. They ensure that an inequality constraint is either active (and then we can require $\alpha_i > 0$) or inactive. The condition in equation A.18 can either be satisfied by $\alpha_i = 0$ which means that the constraint is inactive or by $\alpha_i > 0$ which means that the constraint is active and thus $g_i(\mathbf{w}) = 0$. In the latter case, the constraint behaves like an equality constraint with the difference that we may additionally demand that $\alpha_i > 0$. Therefore, the KKT conditions reflect our intuition that either the optimum of the objective function is in the feasible region of constraint $g_i(\mathbf{w})$ or the constrained optimal point is a point where $-\nabla f(\mathbf{w})$ points away from the feasible region.

A.2 Dual Problem Formulation

In the previous sections we have explained the Lagrange Multipliers method for optimization. Based on this method we can now derive the dual formulation of our SVM optimization problem. Recall that a necessary condition

for a constrained optimum for an objective function (i.e., a solution to the primal problem) satisfies conditions A.15 to A.18.

Now let

$$\overline{L}(\mathbf{w}, \alpha_1, ..., \alpha_n) = \max_{\alpha_1,...,\alpha_n} L(\mathbf{w}, \alpha_1, ..., \alpha_n) = \max_{\alpha_1,...,\alpha_n} \left(f(\mathbf{w}) + \sum_{i=1}^{n} \alpha_i g_i(\mathbf{w}) \right) \tag{A.19}$$

be the maximum of the Lagrange function over $\alpha_1, ..., \alpha_n$ in a given point $\mathbf{w}$. As explained above, Lagrange multipliers for inequality constraints must be at least 0, i.e., $\alpha_1, ..., \alpha_n \geq 0$. Following the argumentation in [28], if all constraints are satisfied at $\mathbf{w}$, the maximum value $\overline{L}(\mathbf{w}, \alpha_1, ..., \alpha_n)$ of the Lagrange function over $\alpha_1, ..., \alpha_n \geq 0$ at this point will be $f(\mathbf{w})$. If, however, at least one constraint k is violated, i.e., if $g_k(\mathbf{w}) > 0$, the maximum will be $\overline{L}(\mathbf{w}, \alpha_1, ..., \alpha_n) = \infty$. Therefore a point $\mathbf{w}$ which minimizes $\overline{L}$ will minimize $f(\mathbf{w})$ if it satisfies all constraints. As a consequence, solving the problem

$$\min_{\mathbf{w}} \overline{L}(\mathbf{w}, \alpha_1, ..., \alpha_n) = \min_{\mathbf{w}} \max_{\alpha_1,...,\alpha_n} L(\mathbf{w}, \alpha_1, ..., \alpha_n) \tag{A.20}$$

for $\mathbf{w}$ and $\alpha_1, ..., \alpha_n \geq 0$ yields a solution to the primal optimization problem. Again $\alpha_i = 0$ marks an inactive constraint while $\alpha_i > 0$ means that the corresponding constraint is active. Since $f(\mathbf{w})$ is convex and the constraints are linear, we can switch the order of minimizing and maximizing:

$$\min_{\mathbf{w}} \max_{\alpha_1,...,\alpha_n} L(\mathbf{w}, \alpha_1, ..., \alpha_n) = \max_{\alpha_1,...,\alpha_n} \min_{\mathbf{w}} L(\mathbf{w}, \alpha_1, ..., \alpha_n) \tag{A.21}$$

Thus, we need to first minimize $L(\mathbf{w}, \alpha_1, ..., \alpha_n)$ over $\mathbf{w}$ and then maximize the result with respect to the Lagrange multipliers $\alpha_1, ..., \alpha_n$.

Since our goal is to derive the dual formulation for the SVM optimization problem, we restate the primal problem formulation. Compared to equation A.2 we re-write the constraints in order to obtain the form $g_i(\mathbf{w}) \leq 0$ which is consistent with the previous explanations.

$$\min_{\mathbf{w},b} \frac{1}{2} \|\mathbf{w}\|^2 \tag{A.22}$$

$$s.t. \;\; \forall i \in \{1, ..., n\} : -\left(y_i(\langle \mathbf{w}, x_i \rangle + b) - 1\right) \leq 0 \tag{A.23}$$

The corresponding Lagrange function is then

$$L(\mathbf{w}, \alpha_1, ..., \alpha_n) = \left(\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} - (y_i(\langle \mathbf{w}, x_i \rangle + b) - 1) \right) \tag{A.24}$$

For minimizing $L(\mathbf{w}, \alpha_1, ..., \alpha_n)$, we compute its derivative $\nabla_{\mathbf{w}} L(\mathbf{w}, \alpha_1, ..., \alpha_n)$ with respect to $\mathbf{w}$ to 0:

$$\begin{aligned}
&\nabla_{\mathbf{w}} L(\mathbf{w}, \alpha_1, ..., \alpha_n) = 0 \\
&\nabla_{\mathbf{w}} \left(\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} - (y_i(\langle \mathbf{w}, x_i \rangle + b) - 1) \right) = 0 \\
&\mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i x_i = 0
\end{aligned}$$

Solving for $\mathbf{w}$ finally yields

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{A.25}$$

We also need to minimize with respect to variable b:

$$\nabla_b L(\mathbf{w}, \alpha_1, ..., \alpha_n) = 0 \tag{A.26}$$

which leads to the equation

$$\sum_{i=1}^{n} y_i \alpha_i = 0 \tag{A.27}$$

Now we substitute A.25 and A.27 in A.24 which yields the minimum of the Lagrange function $\tilde{L}(\mathbf{w}, \alpha_1, ..., \alpha_n)$ with respect to $\mathbf{w}$ and b:

$$\tilde{L}(\mathbf{w}, \alpha_1, ..., \alpha_n) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \left(\sum_{j=1}^{n} \sum_{i=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle \right) \tag{A.28}$$

As explained above (cf. equation A.21), we need to maximize $\tilde{L}(\mathbf{w}, \alpha_1, ..., \alpha_n)$ over $\alpha_1, ..., \alpha_n \geq 0$ which is the dual optimization problem:

$$\max_{\alpha_1,...,\alpha_n} \tilde{L}(\mathbf{w}, \alpha_1, ..., \alpha_n) \tag{A.29}$$

$$s.t. \forall i : \alpha_i \geq 0 \tag{A.30}$$

which implicitly contains all KKT conditions. The dual problem is usually used in practice for multiple reasons: The number of parameters of minimizing the objective function in the primal problem is the length of the training vectors, i.e., the length of model vector $\mathbf{w}$ plus one additional variable b. In the dual problem, one has to minimize over the Lagrange multipliers and thus over as many variables as we have training examples. This is especially beneficial if the model size (length of $\mathbf{w}$) is much larger than the number of training examples which is often the case in practice. The second advantage is that the formulation of the decision function as a scalar product between example vector x with $\mathbf{w}$ translates to scalar products which each training examples (if we substitute equation A.21 in the decision function). This enables the so called "kernel trick" as the scalar product can be replaced by any kernel function which maps the example vectors to a different space where they may be better separable than in their original space. In other words, the kernel trick allows non-linear decision functions, which are, however, beyond the scope of this work. Another related advantage advantage of the dual formulation is the definition of $\mathbf{w}$ which has the Lagrange multipliers as coefficients. Since due to the reasons explained in the previous section, inactive constraints obtain a Lagrange multiplier $\alpha = 0$, many trainings examples can usually be omitted for the final classification function. The remaining training examples with $\alpha > 0$ are commonly called the *support vectors*, hence the name Support Vector Machine. This is especially useful for non-linear kernels, since for the linear kernel discussed in this work, we obviously simply compute the model vector $\mathbf{w}$ and have a single scalar product as decision function.

BIBLIOGRAPHY

[1] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '10, pages 73–80, June 2010.

[2] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202, 2012.

[3] Kobus Barnard, Pinar Duygulu, David Forsyth, Nando de Freitas, David M. Blei, and Michael I. Jordan. Matching words and pictures. *J. Mach. Learn. Res.*, 3:1107–1135, March 2003.

[4] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008.

[5] Jeff A. Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical report, International Computer Science Institute, April 1998.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA, 2006.

[7] Matthew B. Blaschko and Christoph H. Lampert. Learning to localize objects with structured output regression. In *Proceedings of European Conference on Computer Vision: Part I*, ECCV '08, pages 2–15, 2008.

[8] Matthew B. Blaschko, Andrea Vedaldi, and Andrew Zisserman. In John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta, editors, *Proceedings of the Neural Information Processing Systems Conference*, NIPS 23, pages 235–243.

[9] David M. Blei and Michael I. Jordan. Modeling annotated data. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 127–134, New York, NY, USA, 2003.

[10] Chao-Yeh Chen and Kristen Grauman. Watching unlabeled video helps learn new human actions from very few labeled snapshots. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, 2013.

[11] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[12] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1 of *CVPR '05*, pages 886 –893 vol. 1, June 2005.

[13] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[14] Thomas G. Dietterich, Richard H. Lathrop, and Toms Lozano-Prez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(12):31 – 71, 1997.

[15] Santosh Divvala, Alexei Efros, Martial Hebert, and Martial Hebert. Object instance sharing by enhanced bounding box correspondence. In *Proceedings of the British Machine Vision Conference*, pages 60.1–60.11. BMVA Press, 2012.

[16] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[17] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[18] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2 of *CVPR '05*, pages 524 – 531 vol. 2, June 2005.

[19] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, September 2004.

[20] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627 –1645, September 2010.

[21] Thorsten Joachims. Making large-scale svm learning practical. advances in kernel methods - support vector learning. pages 169–184. MIT Press, 1999.

[22] Thorsten Joachims, Thomas Finley, and Chun-NamJohn Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

[23] Michael J. Jones and James M. Rehg. Statistical color models with application to skin detection. *International Journal of Computer Vision*, 46(1):81–96, January 2002.

[24] C.H. Lampert, M.B. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2129 –2142, December 2009.

[25] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of IEEE International Conference on Computer Vision*, volume 2 of *ICCV '99*, pages 1150 –1157, 1999.

[26] Oded Maron and Aparna Lakshmi Ratan. Multiple-instance learning for natural scene classification. In *Proceedings of International Conference on Machine Learning*, ICML '98.

[27] Antonio Monroy and Björn Ommer. Beyond bounding-boxes: Learning object shape by model-driven grouping. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part III*, ECCV'12, pages 580–593, Berlin, Heidelberg, 2012.

[28] Andrew Ng. Lecture notes, http://cs229.stanford.edu/notes/cs229-notes3.pdf. *Stanford University*.

[29] M-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2 of *CVPR '06*, pages 1447–1454, 2006.

[30] Leonid Pishchulin, Mykhaylo Andriluka, Peter Gehler, and Bernt Schiele. Poselet conditioned pictorial structures. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, June 2013.

[31] Christian X. Ries and Rainer Lienhart. Deriving a discriminative color model for a given object class from weakly labeled training data. In *Proceedings of ACM International Conference on Multimedia Retrieval*, ICMR '12, pages 44:1–44:8, 2012.

[32] Christian X. Ries and Rainer Lienhart. A survey on visual adult image recognition. *Multimedia Tools and Applications*, pages 1–28, 2012.

[33] Christian X. Ries, Fabian Richter, and Rainer Lienhart. Towards automatic object annotations from global image labels. In *Proceedings of ACM Conference on International Conference on Multimedia Retrieval*, ICMR '13, pages 207–214, New York, NY, USA, 2013.

[34] C.X. Ries, S. Romberg, and R. Lienhart. Towards universal visual vocabularies. In *Proceedings of IEEE International Conference on Multimedia and Expo*, ICME '10, pages 1067 –1072, July 2010.

[35] Stefan Romberg and Rainer Lienhart. Bundle min-hashing. *International Journal of Multimedia Information Retrieval*, 2(4):243–259, 2013.

[36] Stefan Romberg, Lluis Garcia Pueyo, Rainer Lienhart, and Roelof van Zwol. Scalable logo recognition in real-world images. In *Proceedings of ACM International Conference on Multimedia Retrieval*, ICMR '11, pages 25:1–25:8, New York, NY, USA, 2011.

[37] S. Savarese and L. Fei-Fei. Generic object categorization, localization and pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, ICCV '07, October 2007.

[38] P. Schnitzspan, S. Roth, and B. Schiele. Automatic discovery of meaningful object parts with latent crfs. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 121–128, June 2010.

[39] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, 2001.

[40] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of International Conference on Machine Learning*, ICML '07, pages 807–814, New York, NY, USA, 2007. ACM.

[41] Parthipan Siva, Chris Russell, and Tao Xiang. In defence of negative mining for annotating weakly labelled data. In *Proceedings of European Conference on Computer Vision*, ECCV 2012, pages 594–608, 2012.

[42] J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. In *Proceedings of IEEE International Conference on Computer Vision*, ICCV '03, pages 1470–1477 vol.2, October 2003.

[43] K. Tang, R. Sukthankar, J. Yagnik, and Li Fei-Fei. Discriminative segment annotation in weakly labeled video. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2483–2490, June 2013.

[44] David S. Touretzky, John E. Moody, David S. Touretzky, Gkhan Bakir, Thomas Hofmann, Bernhard Schlkopf, Alexander J. Smola, Ben Taskar, S. V. N. Vishy Vishwanathan, and London England. Predicting structured data, July 2007.

[45] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of International Conference on Machine Learning*, ICML '04, pages 104–, New York, NY, USA, 2004. ACM.

[46] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Secaucus, NJ, USA, 1999.

[47] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Proceedings of IEEE International Conference on Computer Vision*, ICCV '09, pages 606 –613, October 2009.

[48] Paul Viola, John C. Platt, and Cha Zhang. Multiple instance boosting for object detection. In *Proceedings of the Neural Information Processing Systems Conference*, NIPS 18, pages 1419–1426, 2006.

[49] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. HOGgles: Visualizing Object Detection Features. *ICCV*, 2013.

[50] Chun-Nam John Yu. *Improved Learning of Structural Support Vector Machines: Training with Latent Variables and Nonlinear Kernels*. PhD thesis, Cornell University, 2010.

[51] Chun-Nam John Yu and Thorsten Joachims. Learning structural svms with latent variables. In *Proceedings of International Conference on Machine Learning*, ICML '09, pages 1169–1176, New York, NY, USA, 2009. ACM.

[52] A. L. Yuille and Anand Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4):915–936, April 2003.

[53] Long Zhu, Yuanhao Chen, Alan L. Yuille, and William T. Freeman. Latent hierarchical structural learning for object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '10, pages 1062–1069, 2010.